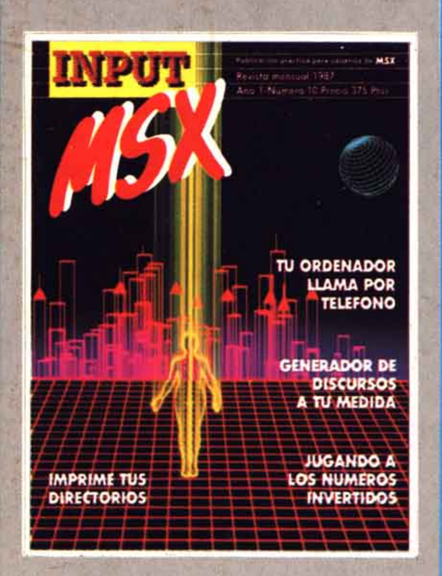

INPUT
Publicación práctica para usuarios de MSX
Revista mensual 1987
Año 1-Número 10 Precio 375 Ptas.
MSX
TU ORDENADOR LLAMA POR TELEFONO
GENERADOR DE DISCURSOS A TU MEDIDA
JUGANDO A LOS NUMEROS INVERTIDOS
IMPRIME TUS DIRECTORIOS



AÑO 1 NUMERO 10

DIRECTOR: Manuel Pérez
COORDINADOR EDITORIAL: Francisco de Molina
DIRECTOR DE ARTE: Luis F. Balaguer
REALIZACION GRAFICA: Didac Tudela
COLABORADORES: Julio García, Xavier Ferrer, Ramón Rabaso, Equipo Molisoft, Javier de la Fuente, Jaime Mardones, Irene Alcaraz, Enric Abad, Antonio Pliego
FOTOGRAFIA: Joan Boada, Ernesto Walfisch

INPUT MSX es una publicación de PLANETA-DE AGOSTINI, S.A.
GERENTE DIVISION DE REVISTAS: Sebastián Martínez

PUBLICIDAD: José Real - Grupo Jota
Madrid: c/ Gral. Varela, 35, 3.º-11
Teléf. 270 47 02/03
Barcelona: Avda. de Sarriá, 11-13, 1.º
Teléf. 250 23 99

FOTOMECANICA: TECFA, S.A.

IMPRESION: Sirven Grafic
C./ Gran Vía, 754-756, 08013 Barcelona
Depósito legal: B. 38.113-1986

SUSCRIPCIONES: EDISA
López de Hoyos, 141. 28002 Madrid
Teléf. (91) 415 97 12

REDACCION:
Aribau, 185, 1.º
08021 Barcelona

DISTRIBUIDORA:
R.B.A. PROMOTORA DE EDICIONES, S.A.
Travesera de Gracia, 56. Edificio Odiseus.
08006 Barcelona.

El precio será el mismo para Canarias que para la Península y en él irá incluida la sobretasa aérea.

INPUT MSX es una publicación controlada por



INPUT MSX es independiente y no está vinculada a los distribuidores del estándar.

INPUT no mantiene correspondencia con sus lectores, si bien la recibe, no responsabilizándose de su pérdida o extravío. Las respuestas se canalizarán a través de las secciones adecuadas en estas páginas.



SUMARIO

UNA GRAN NOTICIA

Este mes ofrecemos una gran noticia a nuestros lectores. A iniciativa de una de las más importantes empresas españolas de software, desde el primer día de marzo los precios de los videojuegos comerciales van a bajar radicalmente su precio.

Por las informaciones de que disponemos en el momento de cerrar este número, la iniciativa ha encontrado una acogida positiva entre el resto de empresas del sector, muchas de las cuales han anunciado su intención de adherirse.

Es obvio que para nuestros lectores ésta no es una novedad cualquiera, pues se trata de un descenso de precios que, en algunos casos, puede situarlos en tan sólo una cuarta parte del que antes tenían en el mercado.

Sin embargo, las consecuencias de esta medida rebasan este aspecto, aun siendo fundamental. Además de acabar con un mecanismo que mantenía los precios artificialmente altos y, por tanto, perjudicaba de forma injustificada al usuario, con las nuevas tarifas el mercado tenderá a ser más dinámico, más receptivo a nuevas ideas e iniciativas y, como consecuencia, a expandirse. Hay que esperar que esta mayor demanda redunde beneficiosamente sobre la producción nacional de software y la industria de comercialización que se genera en torno a ella.

Las nuevas tarifas entran en vigor a partir del mismo día en que esta revista se pone a la venta. Estamos ante una gran noticia para los jóvenes aficionados a los videojuegos y que, en muchos casos, no disponían del poder adquisitivo necesario.

Sin duda, entre usuarios, productores y distribuidores existe conciencia de que la producción interna de software necesitaba de este tipo de medidas para desarrollarse y abandonar su estado de dependencia.

Esperemos que así sea.

LOS MEJORES DE INPUT

Hemos pensado que es interesante disponer de un ***ranking*** que ponga en claro, mes a mes, cuáles son los programas preferidos de nuestros lectores. Para ello, es obligado preguntaros directamente y tener así el mejor termómetro para conocer vuestras preferencias. Podéis votar por cualquier programa aunque no haya sido comentado todavía en **INPUT**.

El resultado de las votaciones será publicado en cada número de **INPUT**.

Entre los votantes sortearemos 10 cintas de los títulos que pidáis en vuestros cupones.

Nota: No es preciso que cortéis la revista, una copia hecha a máquina o una simple fotocopia sirven.

Enviad vuestros votos a: **LOS MEJORES DE INPUT** Aribau, 185. Planta 1. 08021 Barcelona

ELIGE TUS PROGRAMAS

Primer título elegido ____ Segundo título elegido ____

Tercer título elegido ____ Programa que te gustaría conseguir ____

Qué ordenador tienes ____ Nombre ____

1.er Apellido ____ 2.º Apellido ____

Fecha de nacimiento ____ Teléfono ____

Dirección ____ Localidad ____

Provincia ____

INPUT MSX N.º 10

IMPRESION Y LECTURA DE DIRECTORIOS

- LIMITACIONES DE FILES Y DIR
- EL PROGRAMA
- DEFINICION DE VARIABLES
- MENU
- PRESENTACION E IMPRESION

Los afortunados poseedores de unidad de disco sabrán qué fácil es perder toda la información almacenada en un diskette por un mal trato o por no tener protegido éste contra escritura.

Los usuarios de diskettes debemos tener en mente que siempre es preferible contar con una copia de seguridad.

El programador que trabaja en código máquina (o en BASIC) a veces necesita saber qué tiene almacenado en un diskette determinado. Si trabajamos desde el disk-basic el comando FILES nos indicará tan sólo los nombres de los programas almacenados, pero si cargamos el sistema operativo DOS y usamos el comando DIR, además de los nombres de los ficheros obtendremos la longitud (en bytes) y la fecha de entrada en disco de cada uno de ellos. Pero no se nos informa de qué tipo de ficheros son: DOS, BASIC o código máquina. Y si son ficheros de este último tipo no sabemos sus direcciones de inicio, fin y ejecución.

Con el programa que a continuación os presentamos podréis saber (por impresora o por pantalla) el nombre de los programas que hay en el diskette, el tipo de programa, la longitud en bytes de éste y sus direcciones (si es que las posee).

Dando un vistazo al programa, comprobamos que está muy estructurado. A grandes rasgos, consta de seis partes bien diferenciadas.

La primera comienza en la línea 100 y acaba en la 110, y es la que define las variables fundamentales en las que se guardan las características de los ficheros: NN$ (nombre del fichero), LO$ (longitud), DE$ (dirección inicio programa), FI$ (dirección final), ST$ (dirección de ejecución).

La segunda parte, líneas 130-310, es la que muestra el menú por pantalla. Entre las opciones que disponemos están las de lectura de diskette de una o dos caras, pudiendo salir el directorio por la misma pantalla o por impresora (sea MSX o no). Otra opción es la de poder ver un fichero en especial. La última opción es la de regreso al BASIC.

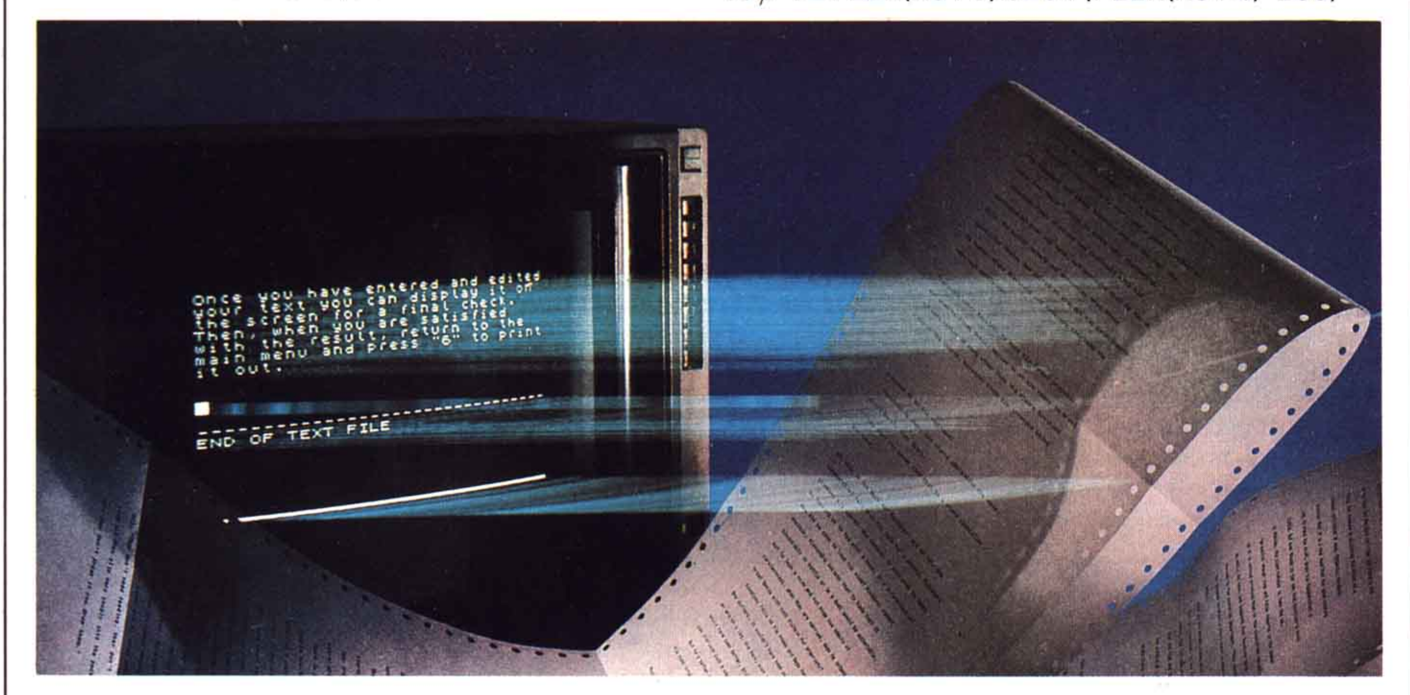
De las líneas 330 y 660 hallamos la estructura básica de lectura (330) y la indicación del tipo de programa: código máquina (500), basic (510), d.o.s (530).

La cuarta y quinta son muy similares: en la cuarta la lectura del diskette se visualiza por pantalla indicando las direcciones de inicio (750), fin (770) y ejecución (790) del programa, si es código máquina, y la longitud en bytes. La quinta parte es quasi-idéntica solamente que sale el directorio por impresora: así las instrucciones PRINT USING se cambian por LPRINT USING (920-1040).

El último bloque consiste en la lectura y salida por pantalla de la información de un fichero en concreto.

```
100 CLEAR5ØØØ
110 DIMNN$(12Ø),NA$(12Ø),LO(12Ø),DE$(12Ø),FI$(12Ø),ST$(12Ø),SE(21Ø)
120 ' Menu principal
130 SCREENØ:KEYOFF:WIDTH38
140 LOCATE3,3:PRINT"****** LECTOR DE DISKETTES *****"
150 LOCATE3,6:PRINT" ┌───┬──────────────────────────┐ "
155 LOCATE3,8:PRINT" │   │                          │ "
160 LOCATE3,7:PRINT" │ 1 │ Lectura directorio 1DD   │ "
165 LOCATE3,10:PRINT"├───┼──────────────────────────┤ "
170 LOCATE3,9:PRINT" │ 2 │ Lectura directorio 2DD   │ "
175 LOCATE3,10:PRINT"├───┼──────────────────────────┤ "
180 LOCATE3,11:PRINT"│ 3 │ Directorio por pantalla  │ "
185 LOCATE3,12:PRINT"├───┼──────────────────────────┤ "
190 LOCATE3,13:PRINT"│ 4 │ Directorio por impresora │ "
195 LOCATE3,14:PRINT"├───┼──────────────────────────┤ "
200 LOCATE3,15:PRINT"│ 5 │ Información de ficheros  │ "
205 LOCATE3,16:PRINT"├───┼──────────────────────────┤ "
210 LOCATE3,17:PRINT"│ 6 │ Finalizar                │ "
215 LOCATE3,18:PRINT"└───┴──────────────────────────┘ "
216 LOCATE3,22:PRINT"******** DIGITE OPCION *********"
```

```
22Ø GOSUB1Ø6Ø
23Ø IFA$<"1"ORA$>"7"THEN22Ø
24Ø IFA$="1"THENSD=5:GOTO33Ø
25Ø IFA$="2"THENSD=7:GOTO33Ø
26Ø IFDI=ØTHENGOTO13ØØ
27Ø IFA$="3"THEN67Ø
28Ø IFA$="4"THEN85Ø
29Ø IFA$="5"THEN1Ø8Ø
3ØØ IFA$="6"THENKEYON:CLS:END
31Ø GOTO22Ø
32Ø ' Lectura del diskette
33Ø AS=PEEK(&HF351)+
    256*PEEK(&HF352):OUT&HAB,
    12:POKE&HFCAB,&HFF
48Ø NEXT K,I:NO=NO-1
49Ø FORI=1TONO:A$=DSKI$(1,
    SE(I)):K=PEEK(AS)
5ØØ IFK=&HFETHENNA$(I)="Maquina":GOTO56Ø
51Ø IFK=&HFFTHENNA$(I)="Basic":GOTO 65Ø
52Ø IFK=&H2ØTHENNA$(I):"Fichero":GOTO 65Ø
53Ø IFK=&HC3THENNA$(I)="MSX-DOS":GOTO
    65Ø
54Ø IFK=&H31THENNA$(I)="Ascii":GOTO 65Ø
55Ø NA$(I)="Desconocido":GOTO 65Ø
56Ø J=PEEK(AS+1):J=J+(PEEK(AS+2)*256)
57Ø A$=HEX$(J):J=LEN(A$)
58Ø DE$(I)=STRING$(4-J,"Ø")+A$
59Ø J=PEEK(AS+3):J=J+(PEEK(AS+4)*256)
```




```
34Ø DI=1:CLS:LOCATE5,12:PRINT"Leyendo el
    directorio..."
35Ø RE=DSKF(1):RE=RE*1Ø24
36Ø NO=1:FORI=SDTOSD+6:A$=DSKI$(1,I)
37Ø FOR K=Ø TO 511 STEP 32:A$=""
38Ø IFPEEK(AS+K)<32THENK=K+32:GOTO48Ø
39Ø IFPEEK(AS+K)>123THENK=K+32:GOTO48Ø
4ØØ FORJ=ØTO7:A$=A$+CHR$(PEEK(AS+K+J))
41Ø NEXT:A$+A$+".":FORJ=ØTO2
42Ø A$=A$+CHR$(PEEK(AS+K+8+J)):NEXT
43Ø NN$(NO)=A$:J=PEEK(AS+K+26)
44Ø J=J*2+SD+3:SE(NO)=J+(PEEK(AS+K+27)*
    512)
45Ø J=PEEK(AS+K+28)
46Ø LO(NO)=J+(PEEK(AS+K+29)*256)
47Ø NO=NO+1
6ØØ A$=HEX$(J):J=LEN(A$)
61Ø FI$(I)=STRING$(4-J,"Ø")+A$
62Ø J=PEEK(AS+5):J=J+(PEEK(AS+6)*256)
63Ø A$=HEX$(J):J=LEN(A$)
64Ø ST$(I)=STRING$(4-J,"Ø")+A$
65Ø NEXT
66Ø GOTO12Ø
67Ø ' Directorio por pantalla
68Ø CLS
69Ø C=Ø
7ØØ C=C+1
71Ø PRINTUSING "                    ";NN$(C);
72Ø PRINTUSING "########";LO(C);
73Ø PRINT" b. ";
74Ø IFNA$(C)<>"Maquina"THEN81Ø
75Ø PRINTUSING"     ";DE$(C);
```

```
760 PRINT" ";
770 PRINTUSING"    ";FI$(C);
780 PRINT" ";
790 PRINTUSING"    ";ST$(C)
800 GOTO830
810 PRINTNA$(C)
820 IFC/20=INT(C/20)THENGOSUB1060
830 IF C<NO THEN 700
840 GOSUB1060:GOTO120
850 ' Directorio por impresora
860 CLS
870 LOCATE8,18:PRINT"Prepare la impresora"
871 LOCATE8,20:PRINT"y pulse una tecla..."
880 GOSUB1060
900 C=0
910 C=C+1:CLS:LOCATE8,10
920 PRINTNN$(C);:LPRINTUSING "    ";NN$(C)
930 LPRINTUSING "#######";LO(C);
940 LPRINT" b. ";
950 IFNA$(C)<>"Maquina"THEN1020
960 LPRINTUSING"    ";DE$(C);
970 LPRINT" ";
980 LPRINTUSING"    ";FI$(C);
990 LPRINT" ";
1000 LPRINTUSING"    ";ST$(C)
1010 GOTO1030
1020 LPRINTNA$(C)
1030 IF C<NO THEN 910
1040 LPRINT:LPRINT" "RE"bytes libres."
1050 GOSUB1060:GOTO120
1060 A$=INKEY$:IFA$=""THEN1060
1070 RETURN
1080 ' Informacion de ficheros
1090 CLS:LOCATE6,2:PRINT"INFORMACION DE
     UN FICHERO"
1100 PRINT:PRINT
1110 LOCATE4,4:PRINT"Pulse la barra
     espaciadora hasta"
1111 LOCATE4,5:PRINT"que aparezca el nombre
     del fichero"
1112 LOCATE4,6:PRINT"deseadoy luego RETURN."
1120 N=0
1130 LOCATE7,18
1140 N=N+1
1150 PRINTNN$(N)
1160 E=0
1170 GOSUB1060
1180 IFASC(A$)<>32ANDASC(A$)<>13THEN
     1170
1190 IFASC(A$)=13THEN1220
1200 IFNO=NTHEN120
1210 GOTO1130
1220 CLS:LOCATE7,9:PRINT"Nombre:   ";NN$(N)
1230 LOCATE7,11:PRINT"Tipo   : ";NA$(N)
1240 LOCATE7,12:PRINT"Longit.: ";LO(N)
1250 IFNA$(N)<>"Maquina"THEN1290
1260 LOCATE7,13:PRINT"Dir.in.: &H";DE$(N)
1270 LOCATE7,14:PRINT"Dir.fin: &H";FI$(N)
1280 LOCATE7,15:PRINT"Dir.ej.: &H"; ST$(N)(N)
1290 LOCATE2,20:PRINT"Pulse una tecla para
     volver al menu":GOSUB1060:GOTO120
1300 CLS:LOCATE2,20:PRINT"Primero debe
     cargar el directorio"
1301 LOCATE2,21:PRINT"del disco en memoria
     del ordenador."
1310 FORN=1TO2000:NEXT:GOTO120
```

GANADORES DE LOS MEJORES DE INPUT MSX

En el sorteo correspondiente al número 15 entre quienes escribisteis mandando vuestros votos a LOS MEJORES DE INPUT han resultado ganadores:

NOMBRE	LOCALIDAD	JUEGO ELEGIDO
Joan Kildal Oquiñena	Andoain (Guipúzcoa)	CAMELOT WARRIORS
Ramón Casas Llobet	Barcelona	SOCCER
José M. Granados Colillas	Sallent (Barcelona)	GREEN BERET
Francisco Belluer Alcántara	Valencia	GREEN BERET
José Luis Pérez González	Madrid	GREEN BERET
David Terrón Panizo	Huesca	THE GOONIES
Fco. Javier Reinaldo Ortega	Viladecans (Barcelona)	GREEN BERET
Albert González Molina	Barcelona	DAMBUSTERS
Marc Campàs Arcarons	Torelló (Barcelona)	JACK THE NIPPER
Luis García Mulet	Oviedo (Asturias)	HERO

SWAPS NUMEROS INVERTIDOS

■ IMPORTANCIA DE LA PRESENTACION
■ EL JUEGO
■ DOMINAR LAS COMBINACIONES

¿Podrás ordenar una serie de nueve números, de menor a mayor, pero invirtiendo sus posiciones en bloques compactos de cuatro o cinco números? Éste es el reto que te plantea SWAPS, el juego de los números invertidos, un rompecabezas que combina lógica y matemáticas, fomentando el desarrollo de la abstracción.

Dada la extensión de este artículo, hemos creído necesario dividir su contenido en dos números de la revista. En esta primera parte, hablaremos de cómo se configura la presentación y del funcionamiento del juego. En el siguiente número publicaremos las rutinas para que podáis poner a prueba vuestra habilidad.

Ahora hablaremos del esqueleto fundamental del programa y de las subrutinas necesarias para realizar la presentación. De esta forma podréis ver la presentación y comprobar, a la vez, que la introducción de esta parte es correcta, en espera del siguiente número.

En el próximo INPUT MSX, explicaremos el funcionamiento del programa, y publicaremos la segunda parte del mismo, con lo cual dispondréis del juego completo.

LA PRESENTACIÓN

Hay una parte del programa en los juegos, a la que, por regla general, no damos la importancia que merece, como es la presentación.

Una carátula bien pensada, con una elaboración cuidada, contribuirá en gran parte al éxito del juego.

Por esta razón, con las magníficas opciones que os brinda vuestro ordenador MSX y un poco de imaginación, también podréis sentiros creativos.

Nosotros, con la presentación que incluimos en nuestro programa, hemos intentado aportar ese granito de arena al diseño, como fiel representante del juego que figura a continuación.

La presentación es como sigue: En primer lugar, aparece una pantalla de un color asalmonado, que va llenándose de números de color negro.

Una vez llena de números que sirven de trama al fondo de la pantalla, se generan unas letras con el nombre del juego, que están degradadas pasando del color amarillo al azul.

A cada letra se le ha asignado una intensidad de color, entre los cinco niveles que hay de un color al otro, puesto que son cinco las letras que componen el nombre **SWAPS**.

Las letras empiezan a moverse, desplazándose por la pantalla y cruzándose entre sí, hasta ocupar cada una el lugar correcto para completar el nombre, quedando, a la vez, perfectamente ordenados los niveles de gradación de color.

Figura unos segundos en pantalla, mientras comienza a desaparecer el fondo gradualmente.

EL JUEGO

El **SWAPS** es un divertido y entretenido juego en el que podéis agudizar vuestro ingenio y habilidad mental.

Se trata de ordenar correlativamente de menor a mayor, una lista de números que va del 1 al 9.

A simple vista parece un juego fácil, ya que sustituir un número por otro no tiene dificultad alguna; pero hay un pequeño inconveniente, y es que el programa no contempla esta opción.

Para manejar los desplazamientos de los números usamos las teclas de función. Cada una de ellas está encargada del movimiento de una zona de la hilera de nueve números.

Las posibles combinaciones son las siguientes:

—La tecla F1 invertirá las cuatro primeras posiciones, es decir, la posición 1, 2, 3 y 4, por posición 4, 3, 2 y 1.

—La tecla F2 invertirá las posiciones 2, 3, 4 y 5, por posición 5, 4, 3 y 2.

—La tecla F3 invertirá las posiciones 3, 4, 5, 6 y 7, por posición 7, 6, 5, 4 y 3. Como veis, giran las cinco cifras, pero la que ocupa la posición central no cambia de lugar.

—La tecla F4 invertirá las posiciones 5, 6, 7 y 8, por posición 8, 7, 6 y 5.

—La tecla F5 invertirá las posiciones 6, 7, 8 y 9, por posición 9, 8, 7 y 6.

Al inicio del juego veremos tres pantallas: En la primera, la máquina os retará a jugar, en la segunda, os dará las instrucciones del juego, y en la tercera, explicará los movimientos de las teclas de función, con ejemplos aclaratorios del funcionamiento.

Durante el juego, en la parte superior de la pantalla aparecerán unos contadores a modo de marcador. En ellos se representan el número de intentos realizados hasta la actual combinación, y el tiempo transcurrido desde el comienzo.

En la pantalla también figurará la posición de las cifras de la jugada anterior, y la presente, representada en otro color que la diferencia del resto.

Cada una de las informaciones que el ordenador nos presenta en la pantalla lo hace dentro de una ventana que simula un cierto relieve sobre el fondo.

Para la solución no hemos establecido ningún límite de intentos ni de tiempo, pero lo que sí podemos asegurar es que el juego siempre tiene solución.

Asegurar la solución es un punto importante en un juego de esta índole, ya que la secuencia de números que aparecen en la pantalla son el resultado de cinco combinaciones de las teclas de función, generadas por el programa, lo que permitiría hallar en cinco combinaciones —si son las co-



rrectas— la colocación ordenada de los números.

En caso de no acertar la combinación ideal, no os deis por vencidos, ya que el programa os brinda como *ayuda* la opción de poderle pedir la solución, partiendo de la cifra obtenida en el último movimiento. Simplemente hay que pulsar la tecla F6, y a continuación la máquina os indicará cuáles son las inversiones a realizar para llegar al final.

Si hacéis uso de esta opción, deberéis retener mentalmente las cifras indicadas, ya que desaparecerán de la pantalla a los pocos segundos.

Si la combinación fuera extremadamente complicada, es decir, que fuera necesario emplear muchas inversiones para llegar a la solución, el microprocesador diría *muy difícil*, negando la ayuda hasta que el jugador mejorase las posiciones de las cifras. Esto significa que el número de inversiones a realizar es tan grande, que supera el espacio que tiene destinado en la pantalla para su impresión.

La ventana donde se representan estos números tiene una capacidad para 16 posibles combinaciones; pero el programa no ignora el resultado, ya que puede retener hasta 256 inversiones. Al mejorar la jugada, si volvéis a pedir *ayuda*, os la dará siempre que la solución sea inferior a 16 movimientos.

Si por error pulsáis una tecla equivocada, simplemente debéis volver a pulsarla, para colocar la combinación de números en la posición anterior, ya que de nuevo volverá a invertir las mismas cifras.

Al finalizar el juego, recibiréis la felicitación del ordenador, que además os informará de en cuántos intentos y tiempo habéis conseguido el triunfo, así como de cuántas ayudas habéis recibido durante la partida.

Si deseáis elevar el grado de complejidad de la solución del juego, en vez de cinco combinaciones de teclas, se puede determinar un número superior de movimientos. Es importante aclarar, que al decir *5 combinaciones de teclas*, nos referimos a que 5 son las inversiones que realiza el programa antes de empezar la partida, no que se pueda ampliar la cantidad de teclas de función.

El programa está estructurado de forma que el usuario pueda alterar ese valor, que viene definido por una constante interna del programa.

Tras el previo estudio del juego, e incidiendo sobre el tema, creemos que al aumentar el nivel, las partidas podrían extenderse excesivamente, tanto en grado de dificultad como en tiempo, pudiendo hacer peligrar el interés hacia el mismo.

Si llegáis a conocer demasiado el juego, de forma que podáis intuir rápidamente la solución, tenéis la posibilidad de acceder al programa, y alterar los cambios asignados a cada tecla de función. De esta manera crearéis una nueva variante sobre el juego.

Al variar la función de las teclas, las partidas siguen teniendo solución, ya que el ordenador confeccionará la lista desordenada, con las combinaciones que halle en las citadas teclas de función.

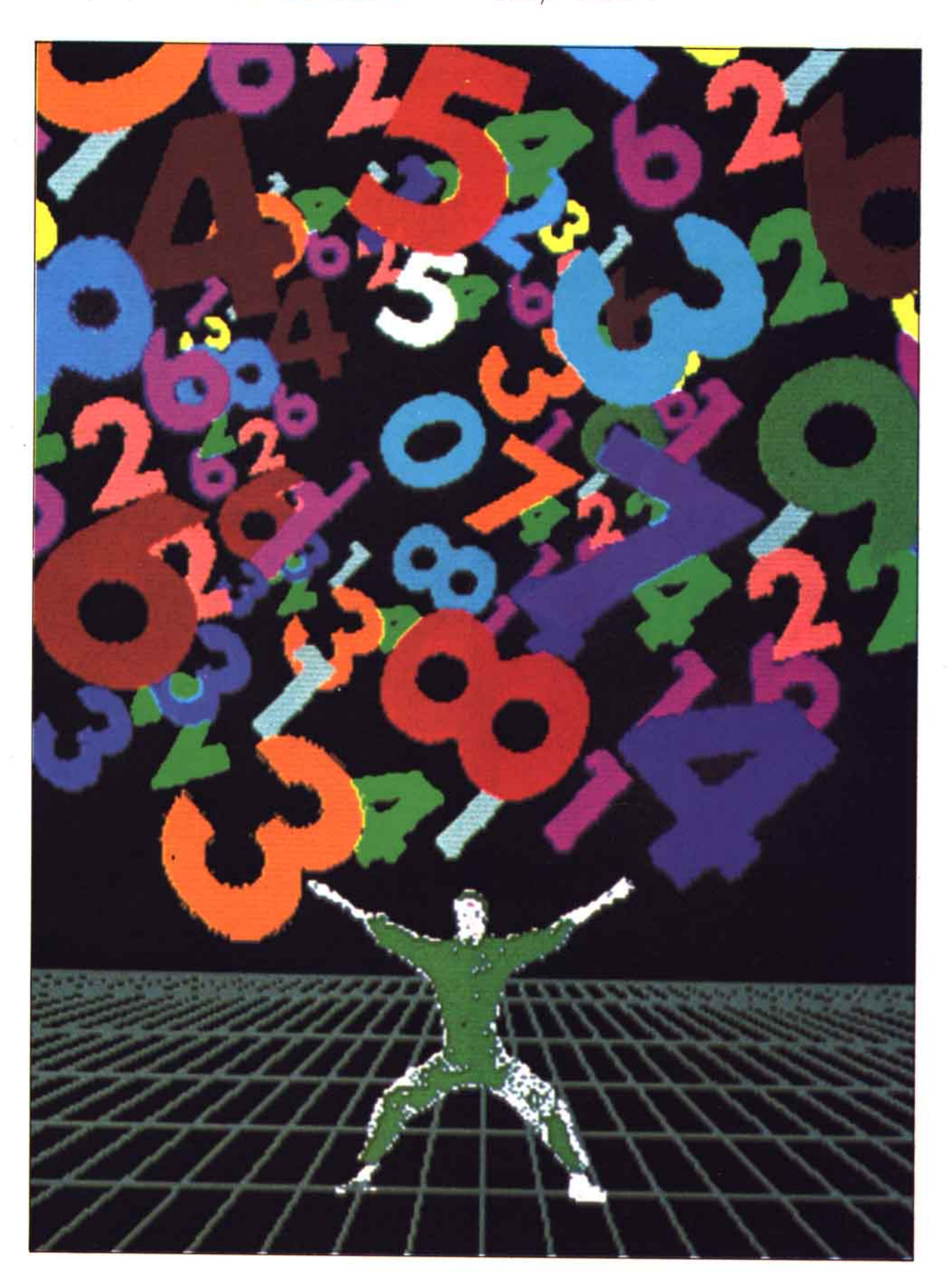
El **SWAPS** no está pensado para un número determinado de jugadores. En caso de ser más de un jugador, debéis ser vosotros mismos quienes confeccionéis unas normas para el juego, como podría ser: fijar un número de intentos o de tiempo, etc., pero esa selección la dejamos en vuestras manos.

Si estáis dispuestos a pasar un rato divertido, aceptad el reto que os hace **SWAPS**.

```
1ØØ '*************************
11Ø '*                       *
12Ø '*    == S W A P S ==    *
13Ø '*                       *
14Ø '*  Juego de los numeros *
15Ø '*       invertidos.     *
16Ø '*                       *
17Ø '*      Por: J.Garci     *
18Ø '*                       *
19Ø '*   REVISTA INPUT MSX   *
2ØØ '*                       *
21Ø '*************************
22Ø CLEAR 4ØØ
23Ø 'Inicializacion
24Ø '--------------
25Ø GOSUB 7ØØ :'sistema 1
26Ø GOSUB 77Ø :'colores
27Ø GOSUB 88Ø :'def sprites
28Ø 'Presentacion
29Ø '------------
3ØØ GOSUB 135Ø:'num fondo
31Ø GOSUB 147Ø:'display letras
```

```
320 GOSUB 1560:'primer
    desplazamiento
330 GOSUB 1690:'segundo
    desplazamiento
340 GOSUB 1890:'firma
350 GOSUB 1990:'borra el fondo
360 'Instrucciones
370 '------------
380 GOSUB 2080:'sistema 2
390 GOSUB 2170:'reto
400 GOSUB 2330:'Instrucciones
405 GOTO 660 :'Fin temporal
410 'Juego
420 '-------------
430 GOSUB 2800:'pantalla juego
440 GOSUB 4380:'Inicio juego
450 GOSUB 3340:'generacion
460 GOSUB 3520:'viejo
470 GOSUB 3640:'nuevo
480 GOSUB 3760:'define teclado
490 GOSUB 4470:'espera jugada
500 GOSUB 3520:'escribe viejo
510 GOSUB 4530:'movimiento
520 GOSUB 4820:'crea solucion
530 GOSUB 4860:'cuenta
    intento
540 GOSUB 3640:'escribe nuevo
550 GOSUB 4950:'ver si fin
560 IF J<9 GOTO 490
570 'Fin de la partida
580 '----------------
590 GOSUB 5020:'puntuacion
600 GOSUB 5210:'otra ?
610 IF A$="s" OR A$="S"
    GOTO 430
620 'Fin del juego
630 '------------
640 GOSUB 5300:'restaura
    maquina
650 CLEAR
660 END
670 ' ===============
680 '=        Subrutinas        =
690 ' ===============
700 'Sistema 1
710 '--------
720 KEY OFF
730 WIDTH 40
740 SCREEN 5,3
750 WIDTH 32
760 RETURN
770 'Colores
780 '-------
790 COLOR=(9,7,3,3):'fondo
800 COLOR=(10,7,7,3):'letras
810 COLOR=(11,6,7,4)
820 COLOR=(12,5,7,5)
830 COLOR=(13,4,7,6)
840 COLOR=(14,3,7,7)
850 COLOR 1,9,9
860 CLS
870 RETURN
880 'definicion de sprites
890 '--------------------
900 'definicion del sprite S
910 '----------------------
920 S1$=CHR$(&HF)+
    CHR$(&H1F)+CHR$(&H3F)
    +CHR$(&H38)
930 S2$=CHR$(&H38)+
    CHR$(&H38)+CHR$(&H3F)
    +CHR$(&H3F)
940 S3$=CHR$(&H1F)+
    CHR$(&HF)+CHR$(&HO)+
    CHR$(&HO)
950 S4$=CHR$(&H20)+
    CHR$(&H3F)+CHR$(&H3F)
    +CHR$(&H3F)
960 S5$=CHR$(&HFC)+
    CHR$(&HFC)+CHR$(&HFC)
    +CHR$(&H4)
970 S6$=CHR$(&HO)+
    CHR$(&HO)+CHR$(&HFO)+
    CHR$(&HF8)
980 S7$=CHR$(&HFC)+
    CHR$(&HFC)+CHR$(&H1C)
    +CHR$(&H1C)
990 S8$=CHR$(&H1C)+
    CHR$(&HFC)+CHR$(&HF8)
    +CHR$(&HFO)
1000 SPRITE$(1)=S1$+S2$+
     S3$+S4$+S5$+S6$+
     S7$+S8$
1010 'definicion del sprite W
1020 '---------------------
1030 W1$=CHR$(&HF8)+
     CHR$(&H70)+
     CHR$(&H70)+
     CHR$(&H70)
1040 W2$=CHR$(&H79)+
     CHR$(&H39)+
     CHR$(&H3B)+
     CHR$(&H3B)
1050 W3$=CHR$(&H3F)+
     CHR$(&H1F)+
     CHR$(&H1E)+
     CHR$(&H1E)
1060 W4$=CHR$(&H1E)+
     CHR$(&HC)+CHR$(&HC)+
     CHR$(&HC)
1070 W5$=CHR$(&H1F)+
     CHR$(&HE)+CHR$(&HE)+
     CHR$(&HE)
1080 W6$=CHR$(&H9E)+
     CHR$(&H9C)+
     CHR$(&HDC)+
     CHR$(&HDC)
1090 W7$=CHR$(&HFC)+
     CHR$(&HF8)+
     CHR$(&H78)+
     CHR$(&H78)
1100 W8$=CHR$(&H78)+
     CHR$(&H30)+
     CHR$(&H30)+
     CHR$
     (&H30)
1110 SPRITE$(2)=W1$+W2$+
     W3$+W4$+W5$+W6$+
     W7$+W8$
1120 'definicion del sprite A
1130 '---------------------
1140 A1$=CHR$(&H7)+
     CHR$(&H7)+CHR$(&H7)+
     CHR$(&HF)
1150 A2$=CHR$(&HE)+
     CHR$(&HE)+CHR$(&H1E)
     +CHR$
     (&H1C)
1160 A3$=CHR$(&H1C)+
     CHR$(&H3F)+
     CHR$(&H3F)+
     CHR$(&H38)
1170 A4$=CHR$(&H78)+
     CHR$(&H70)+
     CHR$(&H70)+
     CHR$(&HF8)
1180 A5$=CHR$(&HE0)+
     CHR$(&HE0)+
     CHR$(&HE0)+
     CHR$(&HF0)
1190 A6$=CHR$(&H70)+
     CHR$(&H70)+
     CHR$(&H78)+
     CHR$(&H38)
1200 A7$=CHR$
     (&H38)+
```

```
      CHR$(&HFC)+
      CHR$(&HFC)+
      CHR$(&H1C)
1210 A8$=CHR$(&H1E)+
      CHR$(&HE)+CHR$(&HE)+
      CHR$(&H1F)
1220 SPRITE$(3)=A1$+A2$+
      A3$+A4$+A5$+A6$+
      A7$+A8$
1230 'definicion del sprite P
1240 '----------------------
1250 P1$=CHR$(&H1F)+
      CHR$(&HF)+CHR$(&HE)+
      CHR$(&HE)
1260 P2$=CHR$(&HE)+
      CHR$(&HE)+CHR$(&HE)+
      CHR$(&HF)
1270 P3$=CHR$(&HF)+
      CHR$(&HE)+CHR$(&HE)+
      CHR$(&HE)
1280 P4$=CHR$(&HE)+
      CHR$(&HE)+CHR$(&HE)+
      CHR$(&H1F)
1290 P5$=CHR$(&HE0)+
      CHR$(&HF8)+
      CHR$(&H3C)+
      CHR$(&H1C)
1300 P6$=CHR$(&H1C)+
      CHR$(&H1C)+
      CHR$(&H3C)+
      CHR$(&HF8)
1310 P7$=CHR$(&HE0)+
      CHR$(&H0)+CHR$(&H0)+
      CHR$(&H0)
1320 P8$=CHR$(&H0)+
      CHR$(&H0)+CHR$(&H0)+
      CHR$(&H0)
1330 SPRITE$(4)=P1$+P2$+
      P3$+P4$+P5$+P6$+
      P7$+P8$
1340 RETURN
1350 'numeros de fondo
1360 '---------------
1370 A$="0573495485390859
      34917562482746328"
1380 A$="A$+"169574548762
      314841578245793651"
1390 K=0
1400 OPEN "grp:" AS #1
1410 FOR Y=1 TO 24
1420 K=1+((K+3) MOD 7)
1430 PRESET (0,Y*8)
1440 PRINT#1,MID$(A$,K,32)
1450 NEXT Y
1460 RETURN
1470 'display letras
1480 '-------------
1490 X1=10:Y1=10:C1=14
1500 X2=60:Y2=10:C2=11
1510 X3=110:Y3=10:C3=12
1520 X4=160:Y4=10:C4=13
1530 X5=210:Y5=10:C5=10
1540 GOSUB 1810
1550 RETURN
1560 'primer desplazamiento
1570 '--------------------
1580 Y=10
1590 FOR I=1
      TO 100
1600 X1=10:Y1=Y
1610 X2=60+I:Y2=Y
1620 X3=110:Y3=Y
1630 X4=160-I:
      Y4=Y
1640 X5=210:Y5=Y
1650 GOSUB 1810
1660 Y=Y+1
1670 NEXT I
```



```
168Ø RETURN
169Ø 'segundo desplazamiento
17ØØ '----------------------
171Ø FOR I=1 TO 1ØØ
172Ø X1=1Ø+(2*I):Y1=Y
173Ø X2=16Ø-I:Y2=Y
174Ø X3=11Ø:Y3=Y
175Ø X4=6Ø+I:Y4=Y
176Ø X5=21Ø-(2*I):Y5=Y
177Ø GOSUB 181Ø
178Ø Y=Y-1
179Ø NEXT I
18ØØ RETURN
181Ø 'colocacion de sprites
182Ø '---------------------
183Ø PUT SPRITE Ø,(X1,Y1),C1,1
184Ø PUT SPRITE 2,(X2,Y2),C2,2
185Ø PUT SPRITE 4,(X3,Y3),C3,3
186Ø PUT SPRITE 6,(X4,Y4),C4,4
187Ø PUT SPRITE 8,(X5,Y5),C5,1
188Ø RETURN
189Ø 'Firma
19ØØ '-----
191Ø COLOR 7
192Ø PRESET (4Ø,1ØØ)
193Ø PRINT #1,"PROGRAMA DE LA REVISTA"
194Ø PRESET (4Ø,13Ø)
195Ø PRINT #1," I N P U T - M S X "
196Ø PRESET (44,16Ø)
197Ø PRINT #1," PLANETA - AGOSTINI"
198Ø RETURN
199Ø 'decolorar el fondo
2ØØØ '------------------
2Ø1Ø FOR K=7 TO Ø STEP -1
2Ø2Ø FOR T=1 TO 2ØØ:NEXT T
2Ø3Ø Q=INT(K/2)
2Ø4Ø COLOR=(9,K,Q,Q)
2Ø5Ø NEXT K
2Ø6Ø FOR T=1 TO 15Ø:NEXT T
2Ø7Ø RETURN
2Ø8Ø 'Sistema 2
2Ø9Ø '---------
21ØØ SCREEN Ø
211Ø WIDTH 4Ø
212Ø COLOR=NEW
213Ø COLOR 15,4,4
214Ø CLS
215Ø KEY OFF
216Ø RETURN
217Ø 'Reto al jugador
218Ø '---------------
219Ø WIDTH 28
22ØØ PRINT"       S W A P S ...."
221Ø PRINT:PRINT:PRINT
222Ø PRINT"Un reto contra los numeros"
223Ø PRINT"desordenados ............"
224Ø PRINT:PRINT:PRINT
225Ø PRINT"Podras ordenar la lista de"
226Ø PRINT"numeros que genero ? ....."
227Ø PRINT:PRINT:PRINT
228Ø PRINT"INTENTALO !"
229Ø T=TIME
23ØØ IF TIME<T+(5*5Ø) GOTO 23ØØ
231Ø CLS
232Ø RETURN
233Ø 'Instrucciones
234Ø '-------------
235Ø CLS
236Ø WIDTH 6Ø
237Ø PRINT" INSTRUCCIONES DE FUNCIONAMIENTO."
238Ø PRINT"================================"
239Ø PRINT
24ØØ PRINT" La maquina genera inicialmente una secuencia"
241Ø PRINT
242Ø PRINT" de nueve numeros desordenados."
243Ø PRINT
244Ø PRINT" Se trata de ordenar la lista en orden creciente"
245Ø PRINT
246Ø PRINT" empleando las teclas de funcion para realizar"
247Ø PRINT
248Ø PRINT" las inversiones siguientes:"
249Ø PRINT:PRINT
25ØØ PRINT"          (Pulsa espacio para continuar)
251Ø A$=INKEY$:IF A$<>" " GOTO 251Ø
252Ø CLS
253Ø PRINT" F1     invierte las cuatro primeras cifras"
254Ø PRINT
255Ø PRINT" 4 3 2 1 5 6 7 8 9 + (F1) = 1 2 3 4 5 6 7 8 9"
256Ø PRINT
257Ø PRINT" F2     invierte las cuatro segundas cifras"
258Ø PRINT
259Ø PRINT" 1 5 4 3 2 6 7 8 9 + (F2) = 1 2 3 4 5 6 7 8 9"
26ØØ PRINT
261Ø PRINT" F3     invierte las cinco cifras centrales"
262Ø PRINT
263Ø PRINT" 1 2 7 6 5 4 3 8 9 + (F3) = 1 2 3 4 5 6 7 8 9"
264Ø PRINT
265Ø PRINT" F4     invierte las cuatro penultimas cifras"
266Ø PRINT
267Ø PRINT" 1 2 3 4 8 7 6 5 9 + (F4) = 1 2 3 4 5 6 7 8 9"
268Ø PRINT
269Ø PRINT" F5     invierte las cuatro ultimas cifras"
27ØØ PRINT
271Ø PRINT" 1 2 3 4 5 9 8 7 6 + (F5) = 1 2 3 4 5 6 7 8 9"
272Ø PRINT
273Ø PRINT" F6     muestra la solucion de la partida"
274Ø PRINT
275Ø PRINT"          (pulsa espacio para iniciar el juego)"
276Ø A$=INKEY$:IF A$<>" " GOTO 276Ø
277Ø CLS
278Ø WIDTH 4Ø
279Ø RETURN
```

ARQUITECTURA DEL ORDENADOR

- NIVEL DE USUARIO
- LENGUAJES DE ALTO NIVEL
- SISTEMA OPERATIVO
- LENGUAJE MAQUINA
- MICROPROGRAMACION

En nuestro artículo anterior hicimos referencia a la historia de los micros. El porqué del nacimiento del Z80, así como una introducción a la estructura del microordenador.

En el presente trabajo vamos a analizar las jerarquías del computador, lo que nos ayudará a aclarar algunos conceptos sobre los niveles que podemos encontrar en un ordenador.

Bueno será comenzar por traer a nuestro recuerdo a aquel *señor de la bata blanca*, que formó parte del importante proceso evolutivo de los ordenadores.

En la década de los 70 había una perfecta dicotomía entre el *Hardware* y el *Software*. Es decir, realizar una operación en apariencia simple, como una división, en lenguaje máquina era una tarea laboriosa para los especialistas, ya que el ordenador no disponía de esas instrucciones y debía ser construida mediante subrutinas.

Este proceso obligaba a que los programas siempre fueran acompañados de una serie de subrutinas, que servían para potenciar el lenguaje máquina que resultaba pobre.

Con los avances tecnológicos llegaron los nuevos microprocesadores, que llevaban ya en el *Hardware* unas instrucciones que facilitaron en gran manera el desarrollo del *Software*, contribuyendo a la desaparición de funciones que hasta entonces se hacían por programa.

Un representante de esta tecnología avanzada es el conocido *Z80*.

El *Z80* es capaz de realizar todas las instrucciones de su predecesor 8080, y además incorpora nuevos códigos de operación antes no definidos, que generan nuevas instrucciones. Algunas de ellas son las que podemos definir como funciones migratorias, ya que para realizarlas con el antiguo 8080 sería preciso emplear un conjunto de ellas formando una subrutina, que ocuparía más espacio en memoria aumentando también el tiempo de ejecución.

Un ejemplo de estas funciones migratorias podría ser la instrucción *LDIR*, que es una instrucción de movimiento de cadenas que se utiliza en gestión de bloques de memorias, así como en gestión de textos.

Antes de su existencia había que hacer un programa que realizara esta función. Ahora, simplemente basta inicializar los parámetros que precise y ejecutar a continuación una sola instrucción.

Si tuviéramos que copiar un bloque de memoria en otra parte de la memoria sólo tendríamos que definir en un registro la dirección de inicio del bloque fuente, en otro, la dirección de inicio del destino, y en un tercero la cantidad de caracteres a mover, ejecutando a continuación la instrucción LDIR.

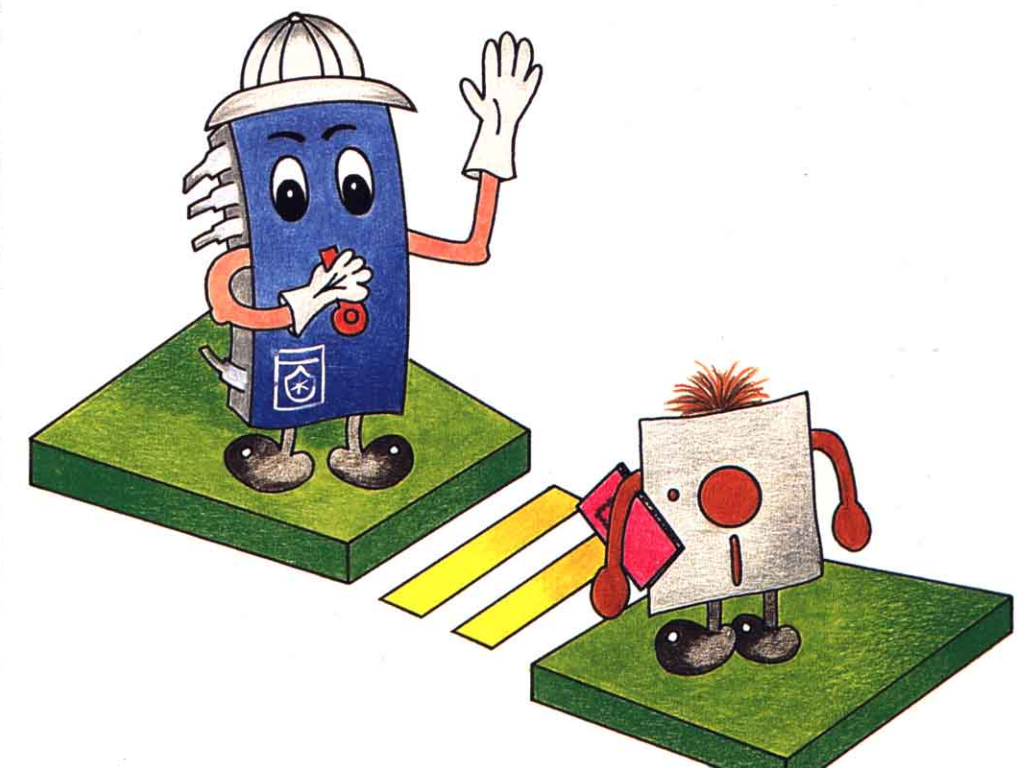
Como hemos citado anteriormente, hasta hace unos años, sólo existían dos niveles, el *Software* y el *Hardware*, y todos los problemas debían resolverse desde estos campos, pero con la sofisticación de los microprocesadores, se han creado otros niveles estratificados, que han dado lugar a lo que llamamos: *Jerarquías del Computador*.

NIVEL DE USUARIO

En el nivel de usuario se encuentran aquellas personas que utilizan los programas del ordenador, pero que en teoría no tienen por qué saber cómo funciona internamente, ni siquiera cómo programarlo. Tan sólo utilizar los programas comerciales, como podrían ser los de gestión de gráficos, procesadores de texto, hojas de cálculo, bases de datos, etc. para aplicaciones concretas, o programas recreativos.

LENGUAJES DE ALTO NIVEL

El segundo escalón corresponde a los lenguajes de alto nivel.



Generalmente los usuarios iniciados programan en estos lenguajes, que consisten en conjuntos de símbolos y palabras que suelen tener como referencia el código ASCII. Algunos lenguajes son: *BASIC, FORTRAN, PASCAL, APL*, etc.

El ordenador MSX se presenta con lenguaje BASIC, que será el utilizado mayormente por los nuevos usuarios, ya que es muy fácil de manejar. También pueden adquirirse en el mercado otros lenguajes e incorporarlos, como puede ser el *PASCAL*.

Los programadores de estos lenguajes no precisan saber cómo está hecho el sistema operativo, ni el lenguaje máquina.

Los microprocesadores sólo pueden ejecutar programas dispuestos en código binario, empleando el *Set* (juego) de instrucciones del lenguaje máquina propio.

Para ejecutar un programa escrito en lenguaje de alto nivel, es preciso traducirlo. Existen dos métodos para efectuar dicha traducción: los lenguajes *compilados*, que son los que traducen el programa antes de ejecutarlo, generando archivos en código objeto (código máquina), ejecutable directamente por la CPU; y los lenguajes *intérpretes*, que son los que traducen cada instrucción en el momento que deben ejecutarla, repitiendo el proceso para cada instrucción.



Generalmente no depende del usuario el poder elegir el tipo de traductor, ya que hay lenguajes de alto nivel, que mayormente son interpretados, como es el caso del BASIC, mientras otros son generalmente compilados, como el PASCAL.

En determinados casos cuando se trabaja con BASIC, es necesario compilar. Como hemos dicho este lenguaje está en intérprete, por lo cual existen algunas máquinas que tienen además del Intérprete BASIC, un Compilador de BASIC.

Entre los lenguajes compiladores e intérpretes existen diferencias importantes en cuanto a ventajas e inconvenientes.

El lenguaje compilador ofrece unas ventajas, como es la velocidad de ejecución, dado que el tiempo de traducción del programa se realiza aparte, propiciando una ejecución más rápida, ocupando menos espacio en memoria y también menos espacio.

Su aplicación resulta ideal en programas que ya están hechos y que por tanto no vayan a modificarse, precisamente por las ventajas comentadas.

Un inconveniente radica en que cuando se compila un programa, y éste tiene errores, debemos corregirlos y volver a compilar de nuevo todo el programa, lo cual hace que resulte lenta la puesta en marcha.

El lenguaje intérprete es ideal para el desarrollo de programas, ya que es mucho más rápido en la puesta a punto de los mismos, pues no hay que compilarlo entero, sino que la parte que se hace funcionar es la que se está interpretando.

El inconveniente del intérprete está en su lentitud, dado que debe ir traduciendo las instrucciones mientras ejecuta el programa.

EL SISTEMA OPERATIVO

El tercer nivel corresponde al sistema operativo. Para trabajar en este nivel no es imprescindible conocer los más bajos, aunque sí aconsejable.

El sistema operativo es el encargado de informar, situar y relacionar el microprocesador con sus periféricos.

Dispone de una tabla de valores, que recibe el nombre de *BIOS* (Basic I/O System), donde se almacenan las direcciones que se refieren a la ubicación de la pantalla, dirección donde se hallan los *floppys*, etc. De esta forma cuando nos dirigimos a un periférico, no debemos preocuparnos de en qué dirección se encuentra, ya que es el Sistema Operativo quien realiza esa función.

En el Sistema Operativo se hallan, además del BIOS, todas las rutinas de gestión de los periféricos del sistema, así como las rutinas aritméticas, trigonométricas y de coma flotante, más otros programas de control del sistema.

Si deseamos realizar alguna gestión de *Entrada/Salida* muy específica desde el lenguaje de alto nivel, podemos efectuar llamadas a las rutinas del S. O., sin necesidad de crear dichas rutinas en lenguaje máquina.

EL LENGUAJE MÁQUINA

El cuarto nivel se refiere al *Lenguaje Máquina*, que es el *Lenguaje Ensamblador*. En este nivel sí debemos conocer a fondo el *Hardware* del ordenador, para saber cómo tiene organizada la memoria, cómo funciona la CPU, etc., ya que vamos a trabajar directamente con el microprocesador.

Este lenguaje es más utilizado en el ámbito profesional, o por los usuarios con conocimientos previos, ya que al trabajar con el microprocesador, el lenguaje es mucho más específico y complicado.

Aunque la elaboración de los programas sea más compleja, tiene ventajas importantes, como son la rapidez de ejecución, y la escasa cantidad de memoria que ocupan, en comparación con otros lenguajes de más fácil acceso para el usuario, como es el BASIC.

Los programas en *Lenguaje Máquina* están codificados en código binario, y se pueden representar mediante sistemas numéricos tales como el binario, el octal y el hexadecimal.

Para llegar a codificar los programas realizados en lenguaje ensamblador, partiremos de los pseudocódigos asociados a cada instrucción del lenguaje máquina, y una vez completos, los tra-



duciremos en códigos binarios. Igualmente hemos de calcular la dirección física absoluta o relativa, para las instrucciones de salto.

Esta tarea de traducción es muy pesada, mecánica y muy presta a cometer errores. Para ayudar a los programadores en *Lenguaje Máquina*, se han creado los denominados *Programas Ensambladores*, que son los encargados de realizar dicha traducción.

Un programa ensamblador generará código máquina, únicamente para el microprocesador para el cual está diseñado.

Al igual que el resto de los lenguajes, se rige por unas normas estrictas, que deben ser respetadas, ya que en caso contrario se generan códigos de error.

La importancia del tema aconseja incidir más ampliamente en un próximo artículo de esta sección, en el que analizaremos la sintaxis del Ensamblador y sus Instrucciones. Mientras tanto os será de utilidad el artículo que publicamos en la sección de Código Máquina de este número.

EL HARDWARE

El siguiente nivel con que nos encontramos es el *Hardware*. En éste es imprescindible conocer todo lo referente al sistema, ya que se trabaja estrechamente con el microprocesador. Debemos saber cuándo va a producirse una selección de memoria, cuándo estará el BUS libre, cuál es la situación de los bits, etc., así como el resto de la circuitería controlada por el Z80, como es el circuito IC TMS-9918, que es el VDP (Video Display Processor), el AY3-8910 que es el PSG (Programable Sound Generator), que es el encargado de la parte de sonido, etc., en una palabra, todo lo referente a su electrónica interna.

LA MICROPROGRAMACIÓN

Existe un nivel, la *Microprogramación*, que sólo existe en los sistemas de ordenadores industriales con las últimas tecnologías.

Podríamos situarlo entre el escalón de *Lenguaje Máquina* y el *Hardware*.

Se trata de un programa, que maneja la electrónica interna de la *CPU* y hace posible que ésta pueda interpretar un determinado *lenguaje máquina*, así sin modificar la circuitería podemos disponer de varios lenguajes máquina para una misma *CPU*.

El Z80 sólo tiene un *Lenguaje Máquina* del que ya hemos hablado anteriormente, ya que su programación en código máquina está cableada, utilizando transistores y circuitos en el interior de la *CPU* que no se pueden cambiar.

En nuestro caso si queremos crear una nueva instrucción, debemos hacer una subrutina que la realice, pero nunca podríamos incluir esta función dentro del conjunto de instrucciones del microprocesador.

Existen potentes microprocesadores, como los de la serie 68000 que son microprogramables por el fabricante, permitiendo que en casos muy concretos se pueda acondicionar directamente el microprocesador a la necesidad específica.

Estas nuevas prestaciones que aporta la *Microprogramación*, facilitan el diseño de las *CPU*, ya que en ese proceso no será necesario pensar cómo será el lenguaje máquina que más tarde utilizará el microprocesador, independizando el diseño electrónico, de la funcionalidad lógica.

El siguiente proceso al diseño, es la fabricación. Será pues el fabricante quien incluirá el microprograma en la *CPU*, la cual con dichas instrucciones mejorará el *Lenguaje Máquina*, que cada vez tenderá a ser más eficiente para la aplicación a que vaya a ser destinado.

Otra versión sería la que nos ofrecen ordenadores como el PDP11, que disponen de una parte de su *Lenguaje Máquina*, modificable por el usuario, ya que el fabricante del ordenador ha dispuesto una pequeña memoria *RAM*, muy rápida y capaz de contener parte del microprograma.

El particularizar el ordenador, implica un pequeño inconveniente, y es que al incluir instrucciones especiales, sólo pueden ser interpretadas por el propio ordenador, y no por otros, que no reconocerían los nuevos comandos.

La microprogramación está justificada en casos, por citar un ejemplo, de cálculos u operaciones que se efectúen con asiduidad y que precisen normalmente el empleo de programas que alargarían el proceso innecesariamente.

Actualmente esta técnica todavía no se ha aplicado al sistema de microordenadores domésticos, pero no descartamos la idea de que se incluya en el futuro próximo.

En el siguiente número realizaremos una descripción exhaustiva de la *CPU*, que nos dará pie para estudiar el *Lenguaje Máquina* del Z80.

AVISO A LOS USUARIOS DE MITSUBISHI

Para aquellos programas que, careciendo de una completa compatibilidad con el standard MSX y MSX.2, presenten problemas de carga en tu ordenador MITSUBISHI ML-G3, ML-G1, ML-FX2 o ML-FX1, ejecuta la siguiente instrucción:

POKE —1,170

Introduce pulsando la tecla «RETURN» y procede a la carga normal del programa.

El uso de esta instrucción es válido para programas cargados desde unidades de disco o cassette indistintamente.

EL ORDENADOR TELEFONICO

El pequeño programa que os presentamos a continuación y que inaugura el concurso de aplicaciones de INPUT MSX, demuestra que es posible marcar un número telefónico a través del ordenador.

¡Ánimo y a enviar vuestras ideas para el concurso!

¿Quién no ha deseado alguna vez ahorrar tiempo al marcar un número telefónico, evitando así la pequeña pero insoportable espera de que el dial vuelva a su posición normal? Esos segundos eran imprescindibles, por ejemplo, para tener ventaja en los concursos telefónicos.

¿Cuántos de nosotros hemos perdido la oportunidad de ganar algún suculento premio, o de tan sólo participar, por ese endemoniado *dedito* que se coló en la cifra que no era, retrasando nuestra entrada en las ondas de la radio?

EL PROGRAMA

El programa aprovecha el relé de que disponen todos los MSX para accionar el motor del cassette a través de los mandatos MOTOR ON y MOTOR OFF.

Comentemos algunos aspectos de dicho programa:

La línea 20 es la encargada de dimensionar la matriz que almacenará las cifras del número de teléfono, el número de cifras (en este caso el 15) podrá ampliarse dependiendo del número de prefijos que tengan que utilizarse en esa llamada.

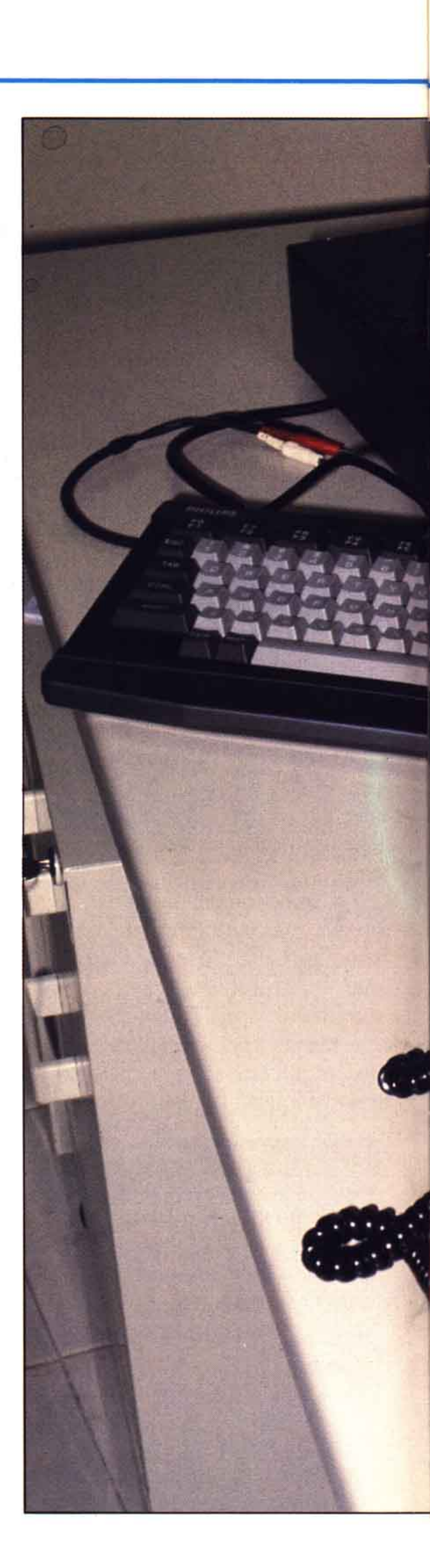
La línea 60 recoge la cantidad de cifras que se quieren marcar y las almacena en la variable N. El bucle FOR...NEXT de las líneas 70 a 100 es el encargado de guardar las cifras del número de teléfono en la variable C. A partir de la línea 110 empieza la parte más importante del programa: primero se conecta la línea telefónica al ordenador (150) esperando este último un cierto tiempo, posteriormente se definen la cantidad de cifras entradas (170), recogiéndose la primera de dichas cifras y conectando y desconectando la línea tantas veces como sea el número escogido. La conexión-desconexión de la línea provocará las pulsaciones que aparecen normalmente al llamar por teléfono.

Las líneas 230 a 250 se encargan de mostrar por pantalla el dígito marcado. Una vez acabada la operación aparecerá un menú con las posibilidades de: llamar de nuevo al mismo número, abandonar el programa regresando al BASIC, o llamar a un número distinto.

Las líneas comprendidas entre 350 y 370 son las encargadas de cambiar el rumbo del programa según la elección del menú anterior. Así, la línea 380 desactiva la línea telefónica llevando el control a 120 donde se repite la operación de marcar el mismo número; la línea 390 desactiva la línea telefónica y finaliza el programa; por último, la 400 nos lleva a marcar una nueva entrada.

En el próximo número de INPUT MSX ampliaremos esta curiosa aplicación. Con ese nuevo desarrollo vuestro ordenador simulará la llamada telefónica con la simple introducción de un nombre de vuestra agenda, gracias a la ampliación del programa con un fichero-agenda.

```
10 SCREEN 0,0,0:COLOR 3,1,
   1:KEYOFF
20 DIM C(15)
30 OPEN"GRP:" AS #1
40 REM MARCADORDE
   NUMEROS
50 REM
60 INPUT "NUMERO DE
   CIFRAS:";N
```



- CONCURSO DE APLICACIONES
- LLAMADA TELEFONICA POR ORDENADOR
- COMENTARIOS AL PROGRAMA




```
70  FOR M=1 TO N
80  PRINT M;:INPUT "-CIFRA:";
    C(M)
90  IF C(M)=Ø THEN C(M)=1Ø
1ØØ NEXT M
11Ø CLS
12Ø SCREEN 2,2,Ø
13Ø Z=5Ø
14Ø LINE (2,98)-(256,1Ø8),8,B
15Ø MOTOR
16Ø FOR G=1 TO 1ØØØ:NEXTG
17Ø FOR M=1 TO N
18Ø FOR R=1 TO C(M)
19Ø MOTOR:FOR
    G=1TO45:NEXTG:MOTOR:FOR
    G=1 TO 45:
    NEXT G
2ØØ NEXT R
21Ø FOR G=1 TO 6ØØ:NEXTG
22Ø IF C(M)=1Ø THEN C(M)=Ø
23Ø PSET (Z,1ØØ),1
24Ø PRINT#1, C(M)
25Ø Z=Z+7
26Ø NEXT M
27Ø FOR M=1 TO N
28Ø IF C(M)=Ø THEN C(M)=1Ø
29Ø NEXT M
3ØØ SCREEN Ø,Ø,Ø
31Ø PRINT "(1) LLAMAR OTRA
    VEZ."
32Ø PRINT "(2) ABANDONAR EL
    PROGRAMA."
33Ø PRINT "(3) LLAMAR A OTRO
    NUMERO."
34Ø PRINT:INPUT "QUE ELIGE";
    F
35Ø IF F=1 THEN GOTO 38Ø
36Ø IF F=2 THEN GOTO 39Ø
37Ø IF F=3 THEN GOTO 4ØØ
38Ø MOTOR:FORG=1TO1ØØØ:
    NEXTG:SCREEN 2,
    2:GOTO12Ø
39Ø MOTOR:END
4ØØ MOTOR:FORG=1TO1ØØØ:
    NEXTG:CLS:GOTO4Ø
```

LISP, EL LENGUAJE DE LAS LISTAS (I)

Aunque empezó con lentitud, el LISP ha ido ganando poco a poco velocidad y las «máquinas de LISP» pueden muy bien ser los prototipos de los ordenadores domésticos que se vendan en los próximos diez a veinte años.

El LISP fue desarrollado originalmente por John McCarthy en el Instituto Tecnológico de Massachusetts a principios de la década de los 60, una época en que los lenguajes de ordenador más sofisticados eran las versiones primitivas del FORTRAN, el lenguaje parecido al BASIC que utilizan ingenieros y científicos. Su nombre se deriva de List Processing, procesado de listas, lo que nos puede dar una pista sobre la forma en que trabaja. Hoy día, veinticinco años más tarde, los programas en LISP continúan basándose en los mismos principios fundamentales, mientras que el FORTRAN ha debido ir incorporando nuevas capacidades a medida que la gente ha ido aprendiendo gradualmente a escribir mejor sus programas.

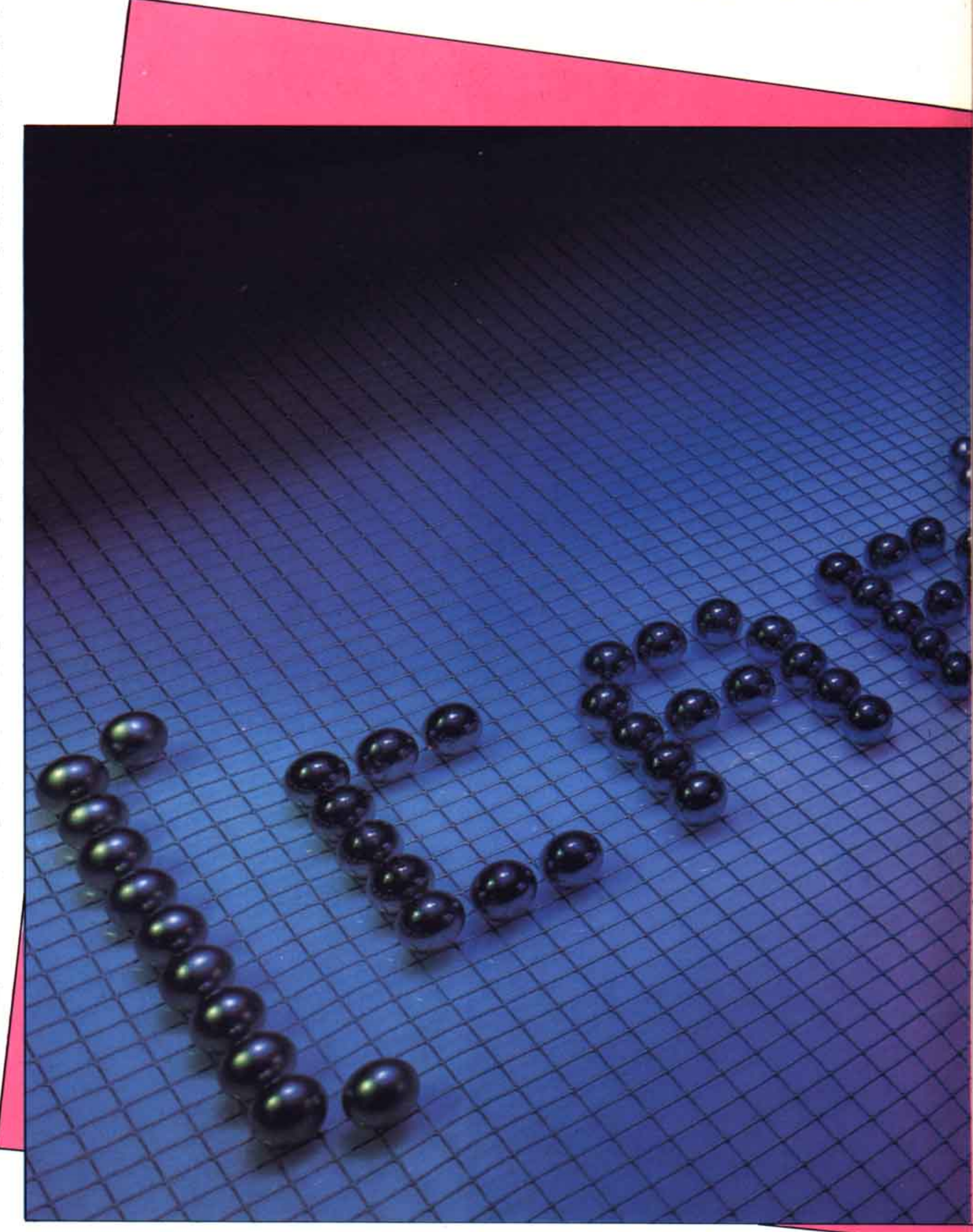
Incluso aunque tú no hayas oído hablar nunca antes del LISP, es muy probable que los programas que escribes se hayan visto influidos por él, debido a que muchos de sus conceptos han sido aplicados a otros lenguajes. No obstante, a pesar de su influencia sobre otros lenguajes, las ideas del LISP siguen siendo radicalmente diferentes de las de lenguajes como el BASIC o el PASCAL, siendo además el LISP un lenguaje considerablemente más potente que ellos.

LA IMPORTANCIA DE LA VELOCIDAD

Es bien conocido que resulta más fácil escribir un programa en BASIC para realizar una determinada tarea, que hacerlo en código máquina. El BASIC es un lenguaje de más alto nivel y sus recursos actúan sobre él como una especie de palanca intelectual haciendo que el programador pueda abordar problemas más difíciles.

En el futuro los ordenadores realizarán tareas mucho más complejas de lo que lo hacen actualmente, lo que significa que los programadores tendrán que utilizar lenguajes aún más potentes, como el LISP, para escribir sus programas; pero los lenguajes más potentes tienden a ser más lentos.

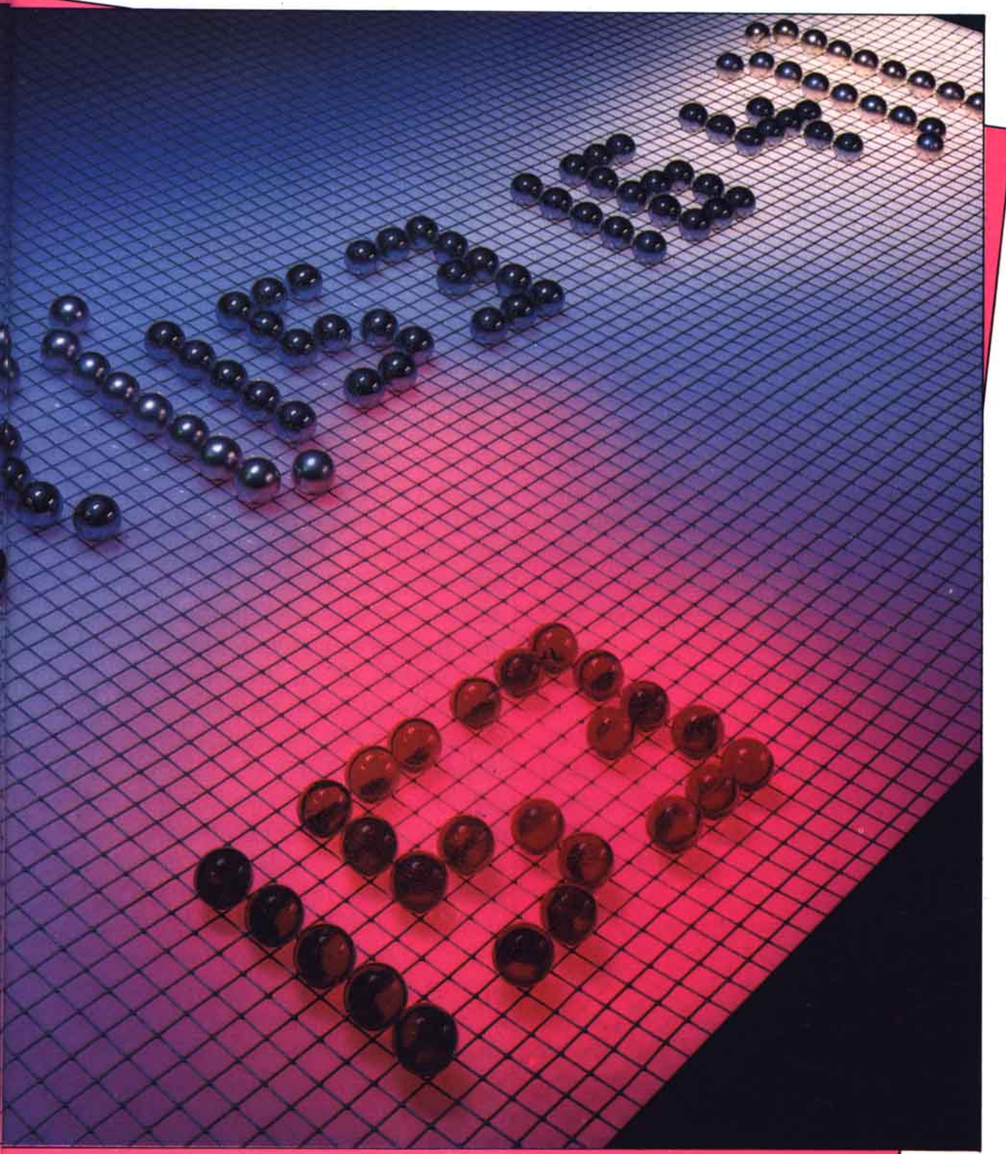
Realmente es cierto que las primeras versiones del LISP eran lentas, lo



cual le quitó popularidad. Sin embargo, las actuales versiones de LISP que corren sobre grandes ordenadores, son tan rápidas como cualquier otro lenguaje de alto nivel. Además el producto de la revolución tecnológica en los circuitos integrados es un enorme crecimiento de la velocidad y capacidad de procesamiento de un solo chip de silicio. Esto significa que ya no resulta vital apurar hasta el último bit la velocidad del sistema utilizando el código máquina.

INTELIGENCIA ARTIFICIAL

El LISP es el principal lenguaje utilizado en Inteligencia Artificial. Su objetivo es aquí la creación de programas de ordenador que presenten un comportamiento similar en algún aspecto a la inteligencia humana. Un ejemplo típico es la máquina que juega al ajedrez. Otro ejemplo es uno de los más famosos en los que se utilizó el LISP. Se trata de un programa llamado ELIZA que desarrolló el papel de psicoanalista, comunicándose con sus pacientes a través de un terminal de ordenador. El programa simulaba la conversación que habría podido tener lugar, preguntando y respondiendo a las preguntas. Hubo personas que realmente se quedaron confundidas pensando que el ordenador entendía de verdad lo que ellas le decían. De hecho lo que hacía el ordenador era simplemente mirar a su entrada y replicar modificándola adecuadamente. El análisis necesario para hacer esto es mucho más fácil en LISP que en otros lenguajes.



Otra área de la inteligencia artificial son los Sistemas Expertos. En este caso se trata de programar en un ordenador todos los conocimientos de un experto humano en algún campo, tal como el mantenimiento de los motores de automóvil. Un usuario sin experiencia puede averiguar lo que le pasa a su coche tecleando los síntomas de los fallos. El ordenador le replica con su diagnóstico, haciendo nuevas preguntas que pueden ser necesarias para localizar con más precisión el problema.

OTRAS APLICACIONES

Otros programas más tradicionales, como compiladores y procesadores de textos, son mucho más sencillos cuando se escriben en LISP, habiéndose utilizado también este lenguaje para escribir los programas de algunos de los sistemas de diseño asistido por ordenador (CAD) utilizados en el diseño de circuitos integrados.

El LISP ha desempeñado un importante papel en los programas de ordenador que se ocupan del álgebra. Es

evidente que los ordenadores son muy buenos en aritmética. Sin embargo con el LISP es posible hacer que lo sean también en álgebra. Esto significa que los ordenadores pueden hacer la clase de cálculos algebraicos que mantendría ocupado a un ejército de matemáticos y llenaría kilómetros de papel.

Algunos de los cálculos de la física teórica contienen un álgebra tan complicada que sólo puede hacerse mediante estos sistemas basados en el LISP. A un nivel más modesto, una aplicación típica del LISP que podrías hacer correr en tu ordenador doméstico podría ser el cálculo de derivadas, lo cual será muy útil si tienes que hacer docenas y docenas de ejemplos de cálculo en tus tareas de casa.

EL LISP Y EL ORDENADOR

Sería erróneo dar la impresión de que el LISP es el único lenguaje capaz de resolver problemas difíciles o de alcanzar relevancia en el futuro. Han aparecido varios lenguajes modernos que utilizan ampliamente sus conceptos, entre los que se incluyen el LOGO y el PROLOG. Con mucha frecuencia los compiladores e intérpretes de estos lenguajes fueron escritos en LISP. Por eso el LISP es en cierta medida el lenguaje ensamblador de la nueva generación de lenguajes de ordenador. Actualmente existen de hecho procesadores especiales cuyos lenguajes ensambladores están basados en el LISP, básicamente de la misma forma que los procesadores Z80, 6502 o 6809 de un ordenador doméstico son capaces de entender el código máquina adecuado.

Estos procesadores de LISP constituyen muchas veces el corazón de determinados ordenadores personales de alta potencia o de puestos de desarrollo de ingeniería, cuyo principal lenguaje es el LISP. Estas «Máquinas de LISP» pueden muy bien ser los prototipos de los ordenadores domésticos que estén a la venta en la calle dentro de cinco o diez años.

En estos artículos sobre LISP, explicaremos los fundamentos del lenguaje y podrás seguir sus desarrollos incluso aunque todavía no tengas un LISP.

Como en el caso del BASIC, existen diferencias entre las versiones del LISP de ordenadores diferentes, aunque afortunadamente las variaciones de uno a otro suelen ser muy pequeñas; normalmente suelen consistir únicamente en diferencias ortográficas en los nombres de las funciones. Un inconveniente del LISP es que hace uso de un elevado número de paréntesis. Si eres una de esas personas que encuentran grandes dificultades para hacer cuadrar el número de paréntesis en las expresiones del BASIC, puede que el LISP no sea el lenguaje más adecuado para ti.

FUNDAMENTOS

Uno de los hechos que distingue al LISP del BASIC es que el LISP, además de números, es capaz de manipular símbolos. Es algo parecido a la forma en que el BASIC puede fraccionar y manejar cadenas de caracteres, si bien la capacidad de manejo de símbolos del LISP es mucho mayor.

Los objetos básicos que maneja el LISP se llaman átomos y listas. Un átomo es realmente como una variable en BASIC, empieza por una letra y puede continuar por cualquier longitud de letras y números (aunque en la práctica siempre hay un límite). Según esto, podrían ser átomos los siguientes: LUIS, atomo, NUM1, NUM2. Todos estamos familiarizados con muchos tipos de listas, por ejemplo una lista de cosas que comprar o la lista de los jugadores de algún equipo deportivo. Tu lista de compras podría ser: té, azúcar y leche. Una lista LISP es algo muy parecido que incluye cierto número de átomos y probablemente otras listas. Para informar al LISP de que algo es una lista, se encierra ésta entre paréntesis.

Así, tu lista de compras en LISP podría ser algo como:

```
(te azucar leche)
```

Una lista de LISP puede contener tanto átomos como otras listas, por lo que otro ejemplo sería:

```
((te frio) (leche caliente) azucar)
```

La anterior lista contiene dos listas y un átomo. Los paréntesis se utilizan de una manera muy parecida a las dobles comillas que usa el BASIC para mostrar dónde empiezan y terminan las cadenas de caracteres. La diferencia consiste en que las cadenas de caracteres no pueden contener otras cadenas de caracteres.

CARGA DEL LISP

Si quieres hacer tus propias pruebas con el LISP, no tienes necesidad de cambiar tu ordenador doméstico, tan hábil con el BASIC, por una máquina de LISP de seis millones de pesetas. De hecho la mayoría de los ordenadores domésticos se pueden convertir al LISP. La única razón por la que un ordenador entiende el BASIC es porque existe un gran programa en ROM que es un intérprete de BASIC. Es perfectamente posible sustituir este código máquina por otro programa que interprete o entienda los programas en LISP. En la práctica, el añadirle el LISP a algunos ordenadores consiste en enchufarles un chip o un pequeño módulo que contiene una ROM que lleva dentro el código máquina necesario. Se suele adoptar con más frecuencia la solución de mantener en la memoria principal del ordenador la sección de código máquina que contiene el intérprete LISP. En tales casos se carga el LISP a partir de una cinta o disco, igual que cualquier otro programa y una vez que el código está dentro de tu ordenador éste se comporta ya como un sistema LISP.

¿Qué es lo que se vería cuando esto sucede? En vez del acostumbrado ok del BASIC, te encontrarías con el mensaje Evaluate:. Esto significa que el LISP está esperando a que teclees alguna expresión válida en LISP, la cual evaluará a continuación e imprimirá su valor en la pantalla. Así es todo el LISP: tú tecleas expresiones y el ordenador te contesta con sus valores. En LISP se llama expresión a una expresión de tipo simbólico que contenga una lista o un átomo. Con frecuencia suele llamarse a esto una s-

expresion. No existe un comando equivalente al RUN que ponga realmente en marcha el programa. Como verás más adelante, este comando no hace falta.

Cuando te aparece el mensaje Evaluate: y el ordenador se queda esperando, tienes que decidir lo que vas a teclear. Lo más sencillo es un átomo, como LUIS. El LISP te responde entonces con un UNDEFINED. El diálogo sería algo parecido a lo que sigue:

```
Evaluate: LUIS
Value is: UNDEFINED
```

Esto significa que el LISP nunca ha visto anteriormente ese átomo y que no puede encontrar su valor. Al igual que las variables del BASIC, los átomos pueden tener valores. La situación se parece a hacer un PRINT de una variable en BASIC antes de asignarle un valor. Cuando el LISP se encuentra el átomo, intenta evaluarlo, es decir, encontrar su valor. Esta es una importante característica del LISP: todo se evalúa siempre a menos de que exista la intención expresa de no hacerlo.

Hay un método para eliminar las evaluaciones en LISP, la comilla sencilla '. Supongamos que tecleas 'LUIS. En estas condiciones, la respuesta del LISP será ahora LUIS. Con la comilla se detiene cualquier evaluación ulterior de lo que viene detrás. El valor de la expresión LUIS es precisamente el átomo que sigue a la coma. También puede usarse la coma con listas. Así:

```
'(te azucar leche)
```

se evaluará como (te azucar leche)

Una aplicación mucho más interesante que la evaluación de átomos es la evaluación de listas. Nuevamente tenemos un ejemplo de esto en la comilla anterior: 'LUIS es realmente una abreviatura de (QUOTE LUIS). De hecho algunos sistemas sólo aceptarán la segunda forma de la expresión. QUOTE es aquí una función del LISP, mientras que el átomo LUIS es su argumento. Observa que QUOTE no halla el valor de su argumento.

Compara esta función con las funciones del BASIC, que podrían ser del tipo:

```
RND (2Ø)
```

El BASIC encuentra para este valor un número aleatorio. Esto mismo en LISP tendría el siguiente aspecto:

```
(RND 2Ø)
```

Observa la nueva situación de los paréntesis. Análogamente, si en BASIC existiera una función llamada QUOTE, se escribiría así:

```
QUOTE (LUIS)
```

Puede que te confunda al principio este nuevo uso de los paréntesis, pero una vez que se ha entendido es sencillo.

VALOR DE LOS ATOMOS

Al igual que las variables, los átomos también pueden tener un valor. Sin embargo surge la cuestión de cómo adquiere un átomo su valor. Por ejemplo, ¿cómo se le asigna a LUIS el valor 1? Se hace con la función SET de la siguiente forma:

```
(SET' LUIS 1)
```

Como de costumbre, el LISP lo evalúa todo. El primer argumento de SET se evalúa como LUIS, y el segundo se evalúa como 1; a continuación la función asigna el valor 1 a LUIS. Esto nos muestra dos cosas. En primer lugar, que en LISP todos los números son átomos que se evalúan para dar su valor numérico. Y en segundo lugar que, igual que ocurre en BASIC, las funciones pueden tener más de un argumento. Si ahora teclearas LUIS, el LISP te respondería con 1. La interacción con el LISP podría ser algo así:

```
Evaluate: (SET' LUIS 1)
Value is: 1
Evaluate: LUIS
Value is: 1
```

Como todas las demás funciones, ha de resultar un valor para SET, normalmente el valor de su segundo argumento que en este caso es 1. Existe otra versión del SET que es SETQ, en la cual no se evalúa el primer argumento. Así, podrías teclear:

```
(SETQ LUIS 1)
```

sin colocar la comilla delante de LUIS. Para insistir aún más en el tema, supongamos que tecleas seguidamente:

```
(SETQ JORGE LUIS)
```

En este caso, el LISP evaluaría LUIS y encontraría que su valor es 1, y a continuación pondría el valor de JORGE a 1. JORGE no se evalúa y no tiene que ser protegido por medio de '.

La función SET se parece al comando LET que tienen algunas versiones del BASIC. Con frecuencia en BASIC las variables tienen que ser de un tipo declarado. En LISP no hay esta necesidad de especificar las variables: un átomo puede tener como valor un número, otro átomo o una lista. Por ejemplo:

```
(SETQ COMPRAS '(te leche
azucar))
```

asignará tu lista de compras al átomo COMPRAS.

Llegados aquí, puede que te hayas dado cuenta de una importante diferencia entre el LISP y el BASIC. Considera en primer lugar la lista:

```
(te leche azucar)
```

es decir, tu lista de compras. Sin embargo, si el té fuera una función (como SET) esta lista podría ser una expresión en LISP y podría ser evaluada. Más adelante veremos que es fácil definir átomos como té para que sean funciones. La cuestión es que en LISP los programas y los datos tienen exactamente el mismo aspecto. Este hecho tiene gran alcance; significa que un programa en LISP puede manejar otro programa en LISP con la misma facilidad con que maneja otros datos.

En LISP todo se hace por medio de

funciones y encontrando el valor de expresiones. Pero a veces esto no resulta muy conveniente. Por ejemplo, normalmente no estarás interesado en el valor real que resulte de la función SET, es mucho más importante que asigne un valor a un átomo. Por eso algunas funciones sólo se utilizan por sus efectos laterales y no por su valor.

DESMENUZANDO LAS PARTES

Para hacer algo con el LISP, evidentemente tienes que poder manipular listas; es algo análogo al manejo de cadenas de caracteres en el BASIC, que las tres funciones fundamentales que hacían esto son CAR, CDR y CONS. La reacción natural ante esta información es «So what?» (¿Qué?). Suponte sin embargo que te dijeran que estas funciones se llaman FIRST, REST y ADD to FRONT. Estas denominaciones te dan realmente mucha más información sobre lo que realmente hacen. De hecho CAR y CDR se han llamado así por ser las iniciales de «contenido del registro de dirección» (contents of address register) y «contenido del registro de decremento» (contents of decrement register), y proceden de los nombres de dos de los registros del ordenador IBM en el que se utilizó inicialmente el LISP. CONS significa construcción. Casi todo el mundo aprende a utilizar estos términos, pero podrían cambiarse fácilmente por ejemplo por (SETQ FIRST CAR).

Tal como sugiere el nombre propuesto como alternativa, CAR da como valor resultante el primer elemento de la lista. Por ejemplo:

```
(CAR '(te leche azucar))
```

da como valor resultante té. Observa que CAR evalúa sus argumentos, por lo que hay que utilizar una ' para evitar la evaluación de la lista. Para ver de otra forma cómo funciona esto, teclea:

```
(SETQ COMPRAS '(te leche
azucar))
```

y a continuación:

```
(CAR COMPRAS).
```

El LISP te respondería de nuevo con té como valor de esto, ya que se encuentra que el valor de COMPRAS es la lista:

```
(te leche azucar)
```

Con CDR resulta la lista que contiene todos los elementos de la lista dada, tomada como argumento, excepto el primero. Así el valor de:

```
(CDR '(te leche azucar))
```

es la lista:

```
(leche azucar)
```

Tal como sugiere su nombre, CONS sirve para construir listas. Hace esto de una manera muy específica, tiene dos argumentos, un átomo o una lista y una lista. CONS agrega su primer argumento al principio de la lista que figura como segundo argumento. Por ejemplo:

```
(CONS 'galletas '(te leche azucar))
```

se evalúa como:

```
(galletas te leche azucar)
```

Para listas, el funcionamiento es el siguiente:

```
(CONS '(izquierdo derecho) '(mano
pie))
```

que tiene el valor:

```
((izquierdo derecho) mano pie)
```

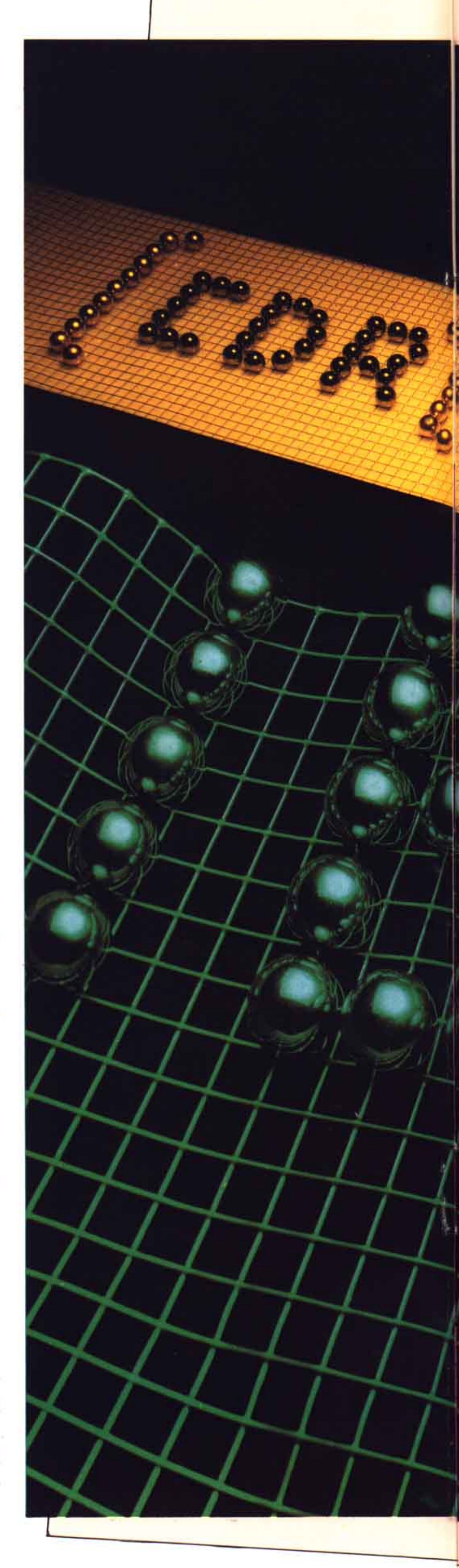
Con CAR y CDR se obtiene en este caso:

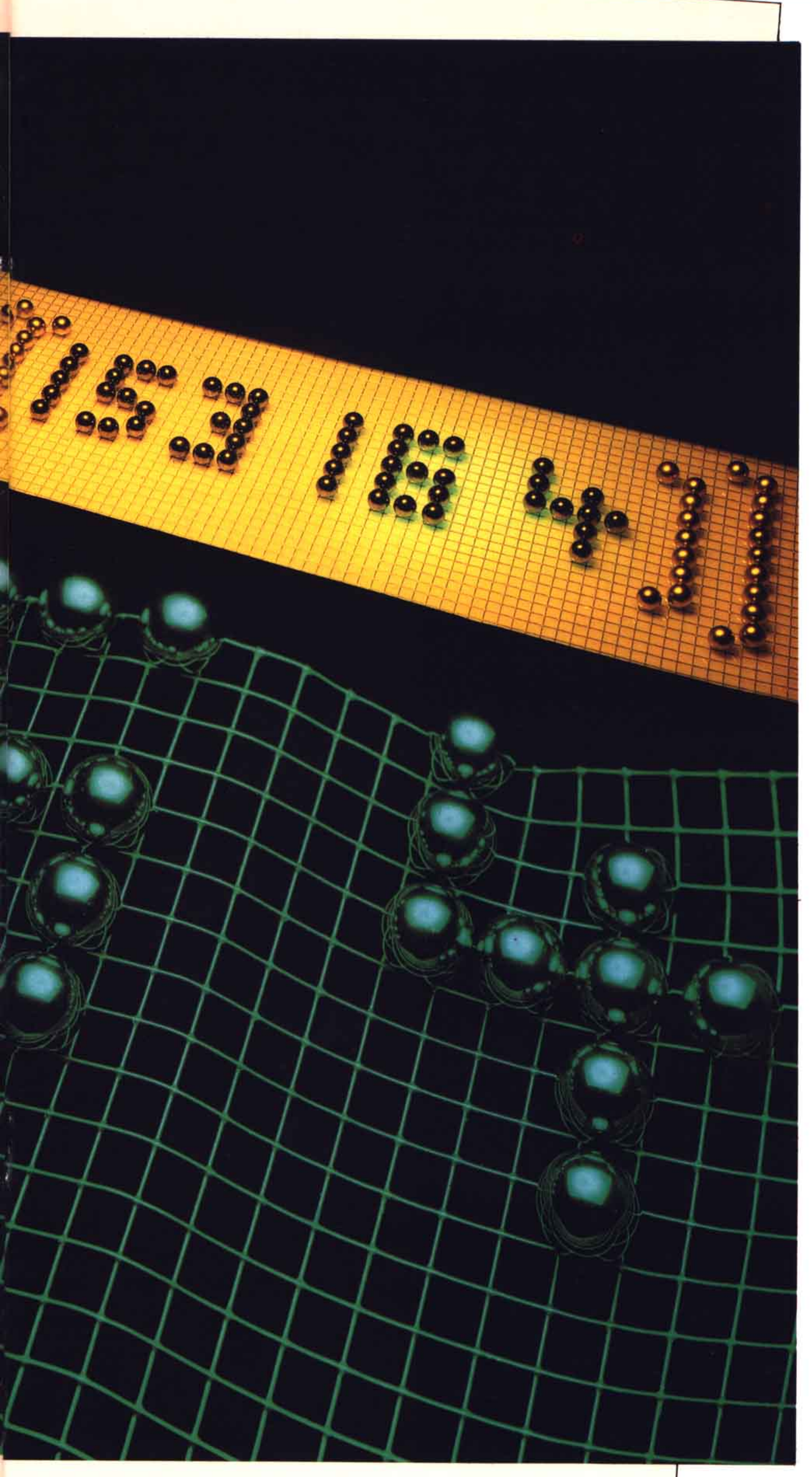
```
(izquierdo derecho)
```

y:

```
(mano pie).
```

Observa que en este caso con CAR se obtiene una lista y no un átomo. Supongamos que tomas el CDR de una lista con sólo un elemento, por ejemplo:





```
(CDR '(LUIS))
```

Esto tiene todo el aspecto de una lista vacía (). De hecho en LISP existe un nombre especial para esto, el átomo NIL. Al tomar el CAR o el CDR de un átomo o de una lista vacía se produce un mensaje de error, tal como era de esperar. Existe una generalización de CAR y CDR. Imagínate que tratas de encontrar el átomo izquierdo de la lista:

```
((izquierdo derecho) mano pie)
```

Podrías encontrarlo como valor de la expresión:

```
(CAR (CAR '((izquierdo derecho)
mano pie)))
```

El LISP te permite hacerlo de una forma más corta con la función CAAR, pudiendo escribirse:

```
(CAAR '((izquierdo derecho) mano
pie))
```

Análogamente, CADR significa tomar el CDR y a continuación el CAR. Los sistemas de LISP permiten también todas las demás combinaciones de CAR y CDR tomados en pares y a veces en ternas, cuaternas y así sucesivamente. Si tuvieras dos átomos LUIS y JUAN y quisieras ponerlos en una lista, podrías sentirte inclinado a teclear:

```
(CONS 'LUIS 'JUAN)
```

Aunque se trata de una expresión válida, no es una lista. Su valor es el siguiente par separado por un punto:

```
(LUIS.JUAN),
```

pero de momento esto es una complicación innecesaria. Una manera de formar la lista deseada sería teclear lo siguiente:

```
(CONS 'LUIS (CONS 'JUAN NIL))
```

El segundo argumento del primer CONS se evalúa para dar la lista (JUAN) en cuyo principio se añade el argumento LUIS. Sin embargo LISP

cuenta también con el método más breve siguiente:

```
(LIST 'LUIS 'JUAN)
```

Este es un ejemplo de otras de las características del LISP: las funciones con un número arbitrario de argumentos. Podrías teclear por ejemplo:

```
(LIST 'UNO 'DOS 'TRES)
```

para obtener la respuesta:

```
(UNO DOS TRES)
```

Esto es muy diferente de lo que ocurre en BASIC, donde las funciones siempre tienen que tener el mismo número de argumentos.

LOS NUMEROS EN LISP

Aunque una de las principales características del LISP es su capacidad de manejar símbolos, también es capaz de manejar números. Las implementaciones del LISP en ordenadores domésticos habitualmente utilizan solamente números enteros. Sin embargo algunos de los grandes sistemas de LISP pueden manejar números en punto flotante y hacer cálculos con ellos con tanta rapidez como cualquier otro lenguaje especial para el cálculo numérico, tal como el FORTRAN. Una peculiaridad que poseen con frecuencia estos sistemas es la aritmética de precisión arbitraria en la que se puede tomar cualquier número de cifras significativas, lo cual resulta perfecto para calcular el valor del número pi con unos cuantos millones de números decimales.

En comparación con el BASIC, el LISP maneja la aritmética de una forma un tanto extraña. Se parece más bien al LOGO, del que ya nos hemos ocupado anteriormente. Los equivalentes de los operadores habituales +, −, *, / y MOD (que da el resto de la división entera) son las funciones PLUS DIFFERENCE, TIMES, QUOTIENT y REMAINDER. La gran diferencia consiste en que el LISP utiliza notación de prefijo, en contraste con el BASIC, que utiliza notación interna. La notación con prefijo se llama a veces notación polaca, en memoria del lógico polaco Lukasiewicz, que demostró que esta notación (así como la polaca inversa, consistente en poner al final el signo de la operación a realizar) son especialmente apropiadas para la aritmética.

Cuando en BASIC quieres sumar 2 y 2, escribes 2 + 2. En LISP esto se convierte en (PLUS 2 2). En la notación con prefijo los operadores aritméticos figuran en primer lugar en las expresiones. Esto permite al LISP tratar a los operadores aritméticos en las funciones exactamente igual que a los demás. Por eso puedes cambiar sus nombres si lo deseas. Algunas de las funciones aritméticas del LISP, como el PLUS, pueden tomar cualquier número de argumentos. Por ejemplo:

```
(PLUS 2 2 2 (TIMES 1 2)
```

se evalúa dando el resultado:

```
2+2+2+1* 2=8
```

En LISP la función MINUS se utiliza para cambiar el signo de un número; por ejemplo el valor de (MINUS 2) es −2. Esto contrasta con el BASIC, donde el símbolo − se emplea tanto para el cambio de signo como para encontrar la diferencia entre dos números.

VERDADERO O FALSO

La siguiente cosa en que tenemos que fijarnos es la forma que tiene el LISP de manejar las expresiones lógicas y cómo hace los saltos condicionales dependiendo del valor de una expresión lógica. En BASIC los valores lógicos se tratan exactamente igual que la aritmética entera; típicamente se utiliza 0 para falso y − para verdadero, aunque son posibles otras elecciones. Una expresión lógica es algo del tipo:

```
A > B
```

Si A es mayor que B, entonces esta expresión en BASIC tiene valor verdadero, en caso contrario tiene valor falso. En LISP verdadero y falso se representan por dos átomos especiales T y NIL (el mismo NIL descrito anteriormente como la lista vacía).

Los valores verdadero y falso resultan de funciones especiales a las que con frecuencia se les llama predicados. Un ejemplo de ello es la función ATOM, que da como resultado el valor T si su argumento es un átomo y NIL en caso contrario. Tecleando:

```
(ATOM 'LUIS)
```

el LISP responderá con el resultado T, mientras que:

```
(ATOM '(leche galletas))
```

dará como resultado NIL. Otros predicados de LISP son GREATERP, para ver cuál de dos átomos tiene mayor valor numérico y NULL que examina una lista para ver si está vacía. El predicado EQ ensaya la igualdad de dos átomos cualesquiera:

```
(EQ'LUIS'COME) (EQ 'LUIS'LUIS)
```

da como resultado T, mientras que:

```
(EQ'LUIS'ENRIQUE)
```

tiene el resultado NIL. En realidad EQ ensaya si sus dos argumentos tienen el mismo valor, así con:

```
(SETQ F '(mermelada pasteles)
```

se tendrá que:

```
(SETQ G F)
```

seguido de:

```
(EQ GF)
```

Tendrá el valor T puesto que los valores resultantes de G y F son la misma lista. Además de éstos, hay muchos otros predicados en LISP. Además de los operadores aritméticos, el LISP cuenta también con el usual complemento de los operadores lógicos. Estos también utilizan la notación de prefijo. Por ejemplo, (OR T T) tiene el valor T, (OR T NIL) tiene el valor T, (AND T NIL) tiene el valor NIL, etc.

En BASIC lo primero que suele hacerse una vez que se está en condiciones de evaluar una expresión lógica es realizar saltos condicionales dependiendo de los valores de dichas expresiones, lo cual se hace con la sentencia IF. El equivalente de esto en LISP se hace por medio de la función COND. Esta función tiene un número arbitrario de argumentos (llamados cláusulas), cada uno de los cuales es una lista que contiene un cierto número de s-expresiones (listas de átomos). Cuando se evalúa la función COND se examinan todas las cláusulas por turno evaluando la primera s-expresión. Si su valor resulta ser T, se examinan todas las s-expresiones subsecuentes de esa cláusula. El valor de la función COND se toma entonces como el de la última s-expresión de la cláusula. En caso contrario la evaluación continúa en la siguiente cláusula. Si ninguna de las cláusulas tiene una primera s-expresión que sea T, la función COND tiene entonces el valor NIL. Normalmente se organizan las cosas de modo que ningún valor NIL para una primera s-expresión dé el mismo resultado que un valor T.

Un caso típico de COND sería el siguiente:

```
(COND  ((EQ LUIS
        1)(PRINT'UNO))
        ((EQ LUIS
        2)(PRINT'DOS))
        ((EQ LUIS
        3)(PRINT'TRES))
```

En este ejemplo se ha introducido otra nueva función, PRINT; su efecto es imprimir el resultado después de evaluarlo. Aunque el LISP cuenta con un conjunto completo de estas funciones de impresión, suelen diferir de un sistema a otro.

Si ahora LUIS ha sido puesto a 2 con SET y se impone la anterior COND, imprimirá la palabra DOS en la pantalla. De hecho el valor de la función PRINT es usualmente el valor de su argumento, por lo que el LISP devolverá el valor DOS como valor de la COND. Está claro por qué sucede esto. En primer lugar el LISP compara el átomo LUIS con 1 y se encuentra con que no son iguales; esto significa que la primera s-expresión de la primera cláusula es NIL. Entonces el LISP se desplaza a la segunda cláusula de la COND. Esta vez la primera s-expresión tiene el valor T, por lo que se evalúa la segunda s-expresión (PRINT 'DOS) con el resultado de imprimir DOS en la pantalla y devolver el valor de la s-expresión (PRINT 'DOS), es decir, DOS.

Todo esto puede parecer bastante complicado y además hay que estar muy atento a los paréntesis en estas expresiones. Sin embargo hay una sentencia muy parecida en el BASIC que hará más claro el funcionamiento de COND. Examina el siguiente fragmento de programa en BASIC:

```
IF LUIS=1 THEN PRINT «UNO»
ELSE IF LUIS=2 THEN PRINT
«DOS»
ELSE IF LUIS=3 THEN PRINT
«TRES»
```

Esta cadena de IF THEN ELSE es equivalente a la función COND. En BASIC estas cadenas terminan muchas veces con un simple ELSE. Esto significa que si no se satisface ninguno de los IFs el último ELSE dominará sobre todos los otros casos. Para tener este comportamiento en LISP, la función COND se termina con la cláusula:

```
(T (código en LISP para otros
casos))
```

Si COND se encuentra con que no se satisface ninguno de sus otros argumentos, llega a esta cláusula y evalúa T que tiene el valor T. Así, las expresiones que siguen a T siempre son evaluadas.

SEGUIMIENTO

Puede que te resulte difícil recordar todo lo que sigue. El LISP te facilita las cosas por medio de una lista de objetos. Dicha lista contiene todos los átomos que el LISP sabe que tienen un valor. La función OBLIST o algo parecido tiene como valor la lista de objetos. Así, para ver todos los objetos conocidos por el LISP, teclea:

```
(OBLIST)
```

Fíjate que aunque esta función no tiene parámetros, también tiene que ir entre paréntesis. En la lista de objetos encontrarás todos los nombres de las funciones familiares como CAR y CDR. También, si haces algo como lo siguiente:

```
(SETQ LUIS 1)
```

te encontrarás el átomo LUIS añadido a la lista.

El hecho clave que permite escribir programas en LISP es, como en el LOGO, que el usuario puede definir sus propias funciones. Estas figurarán entonces en la lista de objetos y, en lo que al LISP respecta, son tan importantes como las que lleva incorporadas el propio LISP. En consecuencia puedes ampliar el LISP para que incluya cualquier cualidad peculiar que quisieras que tuviera; en la segunda parte de esta serie desvelaremos los misterios de cómo usar y definir funciones en LISP.

A NUESTROS NUEVOS LECTORES

En las páginas centrales de la revista encontrarás la sección «Programación de juegos» que se compone de una serie de artículos coleccionables que continúan mes tras mes.

Como la paginación de estos artículos es siempre correlativa con la del mes anterior apreciarás que no se corresponde con la del resto de la revista, pudiendo parecer a los más despistados que faltan páginas o que se trata de un error de encuadernación.

EL GENERADOR DE DISCURSOS

¿Cuántas veces te has visto obligado a dar un discurso? Tal vez en aquella cena con unos amigos, o en la fiesta de cumpleaños en la que eres el protagonista, y claro, evidentemente te ves obligado a decir unas palabras.

Pues bien, se acabó el problema que representa improvisar ese vergonzoso momento, sobre todo si eres una persona no muy dispuesta al monólogo.

Desde ahora con nuestro programa podrás preparar los discursos *a tu medida*. Y cuando decimos *a tu medida*, nos referimos a que podrás hacerlo todo lo extenso que creas oportuno. Basta con decir cuántos párrafos quieres, y el microordenador los seleccionará, escogiendo entre los textos dispuestos en el programa.

Tampoco debe preocuparte sobre qué versará el tema, dado que el texto podemos considerarlo neutro, es decir, si te fijas no estás diciendo nada ni tampoco puedes ofender a nadie: simplemente se trata de un encadenado de palabras que siguen unas normas gramaticales, y que por su construcción y por la utilización de palabras rimbombantes, da la sensación de que hablas de un tema complejo.

En adelante podrás sorprender a tus compañeros, amigos o familiares. Cuando en la próxima ocasión que des un discurso, se encuentren que en vez de aquellas esperadas cuatro frases de siempre, leas tu fabulosa perorata. Si el texto que presenta no es de tu completo agrado podrás fácilmente introducir otras frases que se ajusten más a tus preferencias.

Podrás observar primero su cara de asombro, su perplejidad, y por último el estallido de sus risas, convirtiendo ese momento que antes deseabas evitar en uno de los más divertidos, tanto para ti como para los que te acompañen.

PRESENTACION

El programa a grandes rasgos podemos dividirlo en dos partes: la primera corresponde a los textos, y la segunda al propio programa encargado de organizarlos y mostrarlos en la pantalla y/o impresora.

Como en todo programa de entretenimiento disponemos para comenzar de una presentación. Dicha presentación podemos encontrarla en las instrucciones de las líneas 1140 a 1210, donde nos situamos en SCREEN 3, es decir en modo gráfico de baja resolución, que nos permite hacer grandes y vistosos rótulos.

Recordaréis que en el número 6 de nuestra revista, se dedicó un artículo a *la memoria de Video de MSX: Screen 3*, donde se trató ampliamente el funcionamiento de la Screen 3, a la que ahora nos referimos en la presentación.

Al ejecutar el programa en la pantalla aparecerá el rótulo «prepara tu discurso», que permanecerá durante cinco segundos.

En ese tiempo y a la vez que se ve el rótulo en la pantalla, el microordenador realiza otras tareas como la de cargar los textos en las matrices previamente definidas. Dichos textos forman parte de sentencias DATA almacenadas desde la línea 100 a la 1000, siendo los que más tarde, de forma aleatoria, aunque con cierto control, compondrán el discurso.

Seguidamente espera que acabe de transcurrir el tiempo, y pasa de nuevo a la pantalla 0 con caracteres de tamaño normal, un formato de 80 columnas, papel gris y tinta negra.

Para este programa hemos escogido una presentación sencilla pero vistosa, ya que aunque no se trate de un juego que permita brillantes carátulas llenas de color y movimiento, sí merece una buena portada.

DESCRIPCION DEL PROGRAMA

El comando INPUT de la línea 1460 nos pregunta ¿cuántos párrafos quieres?, debiendo entrar por el teclado, el número deseado.

Asimismo la sentencia de la línea 1490 pregunta ¿salida por la impresora?, poniendo la variable PR a 1 o a 0 dependiendo de si queremos salida por la impresora o no. Estas son las dos únicas preguntas que realiza el programa para poder confeccionar un discurso tan largo como hayamos definido.

En la línea 1540 entramos en lo que hemos llamado la *generación del discurso*, donde se generan todos los párrafos, cada uno con una serie de caracteres de los textos que tiene almacenados en las matrices. Las frases están divididas en cuatro grupos. Para formar un párrafo se toma una frase de cada grupo hasta completar los cuatro.

La línea 250 nos indica que 15 es el número de datos que deben tomarse del primer grupo. De la línea 260 a 400 figura el primer número de datos.

En la 450 vuelve a hacer referencia al número de datos del segundo





grupo, que es también de 15, y así sucesivamente hasta completar los cuatro grupos, que están definidos en la 460 a 600 para el segundo grupo, en la 660 a 800 para el tercero, y en la 850 a 1000 para el cuarto grupo.

Durante la confección del programa, dejamos la opción al usuario de poder ampliar este número de 15 datos hasta 20, dando versatilidad al mismo. Así creamos la matriz DM(4, 20), es decir cuatro, una por cada grupo, y 20, porque puede tener hasta 20 frases aunque en este momento disponga de 15.

COMBINACION ALEATORIA

Al ir apareciendo los textos de forma aleatoria, podría darse el caso de que se repitieran algunos de ellos con una frecuencia inadecuada. Para evitar este problema, en la línea 1320, se ha definido la variable LT a 5, que es el número de párrafos que deben transcurrir hasta volver a permitir la utilización de la misma frase. Por ejemplo, si modificamos la variable LT a 1, permitimos que las frases puedan repetirse constantemente. De esta forma aumentando o disminuyendo el valor, variaremos la frecuencia máxima de repetición de las frases.

Para su control existe un contador que inspecciona los párrafos y que im-

pide a la frase que aparezca antes de pasar los cinco párrafos, a partir de ese momento la frase queda libre pasando de nuevo a estar disponible.

Otro de los problemas que podrían plantearse es la aleatoriedad. El microordenador dispone ya de un sistema de números aleatorios, pero no era apropiado para nuestro cometido, ya que tiene tendencia a comenzar siempre por los mismos números, motivo por el cual creamos otro sistema, que encontramos en la línea 1540, y que está basado en el tiempo de funcionamiento del ordenador.

Con estos condicionantes llegamos a la línea 1630, donde se analiza si la frase seleccionada está prohibida o no. En caso de que lo esté, busca otra.

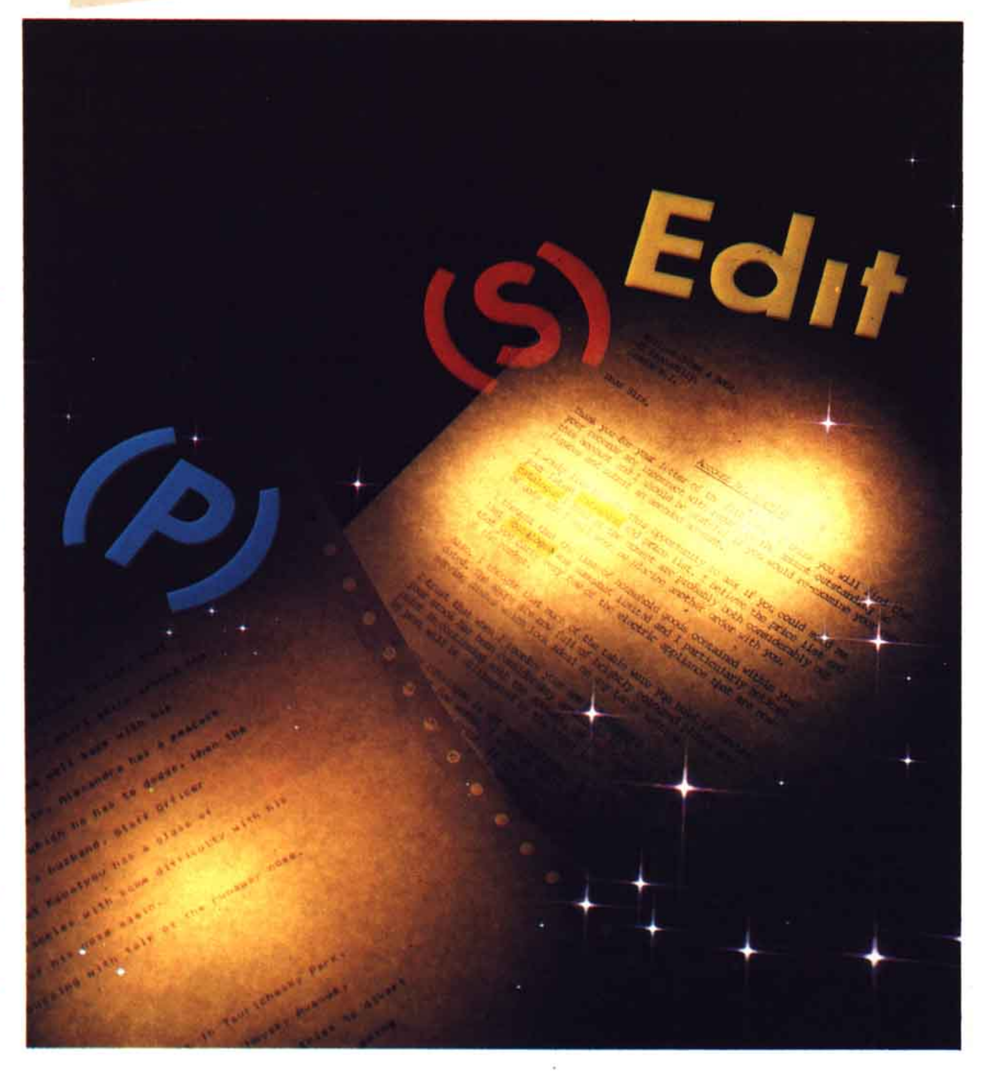
Una vez tiene la frase permitida, pasa a llamar a la subrutina de «escritura sin romper palabras», situada en la línea 1800. Como su nombre indica esta subrutina se encarga de revisar el final de línea de las frases, teniendo como objeto que ninguna palabra quede cortada. Para ello, mira el final buscando un espacio; si no lo encuentra, va quitando caracteres hasta localizarlo, pasando la última palabra a la línea siguiente.

Hemos definido un ancho de línea de 70 caracteres, aunque este número puede ser modificado por el usuario, rectificando la variable LN de la línea 1450.

Esta subrutina que va de la línea 1800 a 1970, podría emplearse aisladamente de este programa, en otros programas, siempre que deseemos imprimir palabras enteras.

Cuando ha terminado el número de párrafos preseleccionados, pregunta ¿quieres otro«, esto sucede en la línea 1730.

Al terminar el microordenador vuelve a restaurar las condiciones normales de la máquina.



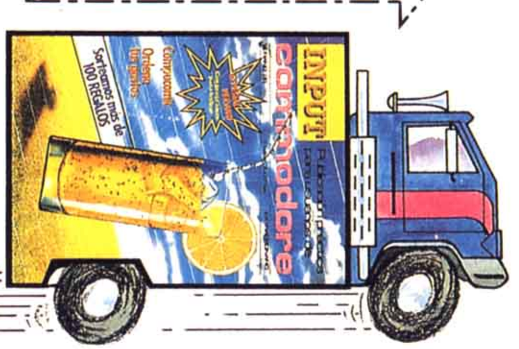

```
100 ' ***********************
110 ' *                     *
120 ' * Programa generador de *
130 ' *                     *
140 ' *       discursos de   *
150 ' *                     *
160 ' *       compromiso.    *
170 ' *                     *
180 ' *       INPUT MSX     *
190 ' *                     *
200 ' ***********************
210 '
220 'Primer grupo de datos.
230 '--------------------
240 '
250 DATA 15
260 DATA "Queridos colegas, "
270 DATA "Por otra parte, "
280 DATA "Asi mismo, "
290 DATA "Sin embargo, no
    olvidemos que "
300 DATA "De igual manera "
310 DATA "La practica de la vida
    cotidiana prueba que "
320 DATA "No es indispensable
    argumentar el peso y la
    significacion de estos
    problemas ya que "
330 DATA "Las experiencias ricas
    y diversas, "
340 DATA "El afan de
    organizacion, pero sobre todo
    "
350 DATA "Los principios
    superiores tecnologicos, asi
    como "
360 DATA "No olvidemos que "
370 DATA "Sin lugar a dudas,
    podemos afirmar que "
380 DATA "No es mi intencion
    extenderme demasiado en el
    tema, pero debemos decir
    que "
390 DATA "Hemos realizado
    profundos estudios, que nos
    han llevado a la conclusion
    de que "
```

```
400 DATA "Asi pues, amigos
    mios "
410 '
420 'Segundo grupo de datos.
430 '----------------------
440 '
450 DATA 15
460 DATA "la realización de los
    deberes del programa "
470 DATA "la complejidad de los
    estudios realizados por los
    dirigentes "
480 DATA "el aumento constante
    en cantidad y en extensión
    de nuestra actividad "
490 DATA ''la estructura actual
    de la organizacion''
500 DATA "el nuevo modelo de la
    actividad de la organizacion "
510 DATA "el desarrollo continuo
    de distintas formas de
    actividad "
520 DATA "la garantia constante
    de nuestra actividad de
    informacion y propaganda "
530 DATA "el reforzamiento y
    desarrollo de las estructuras
    "
540 DATA "la consulta con
    numerosos colaboradores y
    simpatizantes "
550 DATA "el inicio de la accion
    general de formacion de las
    actitudes "
560 DATA "la modificacion
    realizada en la estructura de
    la informacion "
570 DATA "la creacion de
    proyectos complejos conlleva
    un inevitable riesgo que "
580 DATA "el modelo de creacion
    actualmente existente "
590 DATA "la modernizacion de
    las tendencias actuales "
600 DATA "la linea que define
    nuestra actuacion "
610 '
620 'Tercer grupo de datos.
630 '---------------------
640 '
650 DATA 15
660 DATA "nos obliga al analisis"
670 DATA "cumple un rol
    esencial "
680 DATA "exige la precision y la
    determinacion "
690 DATA "ayuda a la
    preparacion y a la realizacion
    "
700 DATA "garantiza la
    participacion de un grupo
    importante en la formacion "
710 DATA "cumple deberes
    importantes en la
    determinacion "
720 DATA "facilita la creacion "
730 DATA "no obstaculiza la
    apreciacion de la
    importancia "
740 DATA "ofrece un ensayo
    importante de verificacion "
750 DATA "implica el proceso de
    reestructuracion y de
    modernizacion "
760 DATA "colabora a la mayor
    organizacion en el estudio "
770 DATA "incorpora nuevas
    directrices, siempre
    interesantes en la creacion "
780 DATA "aporta entusiasmo
    para continuar, cada vez con
    mas fuerza, con la tarea de
    definicion "
790 DATA "nos obliga a preparar
    unas bases definitivas para
    el desarrollo "
800 DATA "conlleva la
    reestructuracion dinamica "
810 '
820 'Cuarto grupo de datos.
830 '---------------------
840 '
850 DATA 15
860 DATA "de las condiciones
    financieras y administrativas
    existentes "
870 DATA "de las directivas de
    desarrollo para el futuro "
880 DATA "del sistema de
    participacion general. "
890 DATA "de las actividades de
    los miembros de las
    organizaciones hacia sus
    deberes."
900 DATA "de las nuevas
    proposiciones "
910 DATA "de las direcciones
    educativas en el sentido del
    progreso. "
920 DATA "del sistema de
    formacion de cuadros que
    corresponda a las
    necesidades "
930 DATA "de las condiciones de
    las actividades apropiadas. "
940 DATA "del modelo de
    desarrollo. "
950 DATA "de las formas de
    accion. "
960 DATA "de los marcos en los
    que se desarrolla la
    actividad."
970 DATA "de bases previas a los
    acuerdos que determinan
    nuestro futuro."
980 DATA "de la creacion de
    nuevas normas."
990 DATA "de las actividades."
1000 DATA "de las nuevas ideas
     que fomentan a la vez a los
     nuevos proyectos."
1010 '
1020 'Inicializaciones.
1030 '----------------
1040 KEY OFF
1050 SCREEN 3
1060 OPEN "grp:" AS #1
1070 KEY OFF
1080 COLOR 10,5,5
1090 CLS
1100 DIM T$(4,15)
1110 DIM N(4)
1120 DIM DM(4,20)
1130 PR=0:'printer off'
1140 'Titulo.
1150 '-------
1160 PRESET (10,10)
1170 PRINT #1, "Prepara "
1180 PRESET (10,60)
1190 PRINT #1," tu "
1200 PRESET (10,110)
1210 PRINT #1,"discurso"
1220 'Asignacion de cadenas
1230 '--------------------
1240 TIME=0
1250 RESTORE
```

```
1260 FOR I=1 TO 4
1270 READ N(I)
1280 FOR J=1 TO N(I)
1290 READ T$ (I,J)
1300 NEXT J
1310 NEXT I
1320 LT=5:'Limit para repet.
1330 FOR J=1 TO 4
1340 FOR I=1 TO 20
1350 DM (J,I)=LT=1
1360 NEXT I
1370 NEXT J
1380 IF TIME<50*5 GOTO 1380
1390 'Entrada de datos.
1400 '-----------------
1410 SCREEN 0
1420 WIDTH 80
1430 COLOR 1,14,14
1440 CLS
1450 LN=70:LC=0
1460 INPUT "Numero de parrafos "; NP
1470 NP=INT(ABS(NP))
1480 IF NP=0 THEN PRINT "Como que ninguno, por lo menos uno no ?"
1490 INPUT "salida por impresora (S/N)"; PT$
1500 IF PT$="s" THEN PT$="S"
1510 IF PT$="n" THEN PT$="N"
1520 IF PT$<>"S" AND PT$<>"N" GOTO 1490
1530 IF PT$="S" THEN PR=1 ELSE PR=0
1540 'Generacion del discurso.
1550 '------------------------
1560 A=TIME
1570 A=A-(10*INT(A/10))
1580 K=RND(-A)
1590 PRINT:PRINT
1600 FOR I=1 TO NP
1610 FOR J=1 TO 4
1620 N1=1+(INT(RND(A)*N(J)))
1630 IF DM(J,N1)<LT GOTO 1620
1640 DM (J, N1)=0
1650 FOR K=1 TO N(J)
1660 DM(J,K)=DM(J,K)+1
1670 NEXT K
1680 A$=T$(J,N1):GOSUB 1800
1690 NEXT J
1700 GOSUB 1980: GOSUB 1980
1710 NEXT I
1720 PRINT:PRINT
1730 PRINT "Deseas otro discurso (S/N)"
1740 A$=INKEY$
1750 IF A$="" GOTO 1740
1760 IF A$="S" OR A$="s" GOTO 1390
1770 IF A$="N" OR A$="n" GOTO 2090
1780 IF PR=1 THEN LPRINT P$;
1790 GOTO 1730
1800 'escritura sin romper palabras
1810 '----------------------------
1820 IF LC+LEN(A$)<LN GOTO 1940
1830 LP=LN-LC+1
1840 IF MID$(A$,LP,1)=" " GOTO 1880
1850 LP=LP-1
1860 IF LP=0 GOTO 1900
1870 GOTO 1840
1880 P$=LEFT$(A$,LP)
1890 GOSUB 2040:'print p$
1900 GOSUB 1980:'print
1910 LC=0
1920 A$=RIGHT$(A$,LEN(A$)-LP)
1930 GOTO 1820
1940 LC=LC+LEN(A$)
1950 P$=A$
1960 GOSUB 2040:'print p$
1970 RETURN
1980 'salto de linea
1990 '--------------
2000 PRINT
2010 IF PR=1 THEN LPRINT
2020 LC=0
2030 RETURN
2040 'escritura de p$
2050 '--------------
2060 PRINT P$;
2070 IF PR=1 THEN LPRINT P$;
2080 RETURN
2090 'fin de programa
2100 '--------------
2110 SCREEN 0
2120 COLOR 15,4,4
2130 WIDTH 40
2140 CLS
2150 KEY ON
2160 END
```

LA REDACCION CAMBIA DE DIRECCION





Aribau n.º 185 planta, 1 08021 Barcelona

BARAJA Y REPARTE

- DISPOSICION DE LAS CARTAS
- DIBUJANDO LAS CARTAS
- BARAJADO
- REPARTO

Los ordenadores pueden ser muy buenos jugadores de cartas si se programan correctamente; además nunca se aburren.

Aquí tienes la forma de programar los gráficos de una baraja.

¿Te encuentras distanciado de tus amigos, parientes o colegas por haberles dejado sin un duro jugando a las cartas? ¿Eres tú el que está sin blanca por haber jugado con expertos? Sea como fuere en los capítulos siguientes de nuestro coleccionable te presentamos la solución. Programando tu ordenador para que juegue contigo a las veintiuna, tendrás una víctima propiciatoria.

En esta primera parte nos ocuparemos de la manera de generar las rutinas gráficas con las que se construye la baraja. el resto del programa que es el juego propiamente dicho se presentará en los dos capítulos siguientes.

Pero no te olvides de almacenar cada una de las secciones en cinta a medida que construyes el juego.

Si no eres un experto en el juego de las veintiuna, no te preocupes: en la última parte del programa presentaremos un conjunto completo de las reglas del juego pero antes debes ser capaz de programar un mazo de cartas.

La rutina que te permitirá barajar las cartas y preparar las rutinas gráficas sería algo así:

```
10 CLEAR 400
20 DIM C(52),P$(52),N(52),
   V$(52),W(52)
30 OPEN "grp:" AS #1
40 REM CRUPI EL CRUPIER
50 CLS:KEYOFF:COLOR 3,1,1
130 COLOR 3,1,1
150 SCREEN 2
160 LINE (10,10)-(245,150),
    10,B
170 PAINT (5,5),10
180 LINE (10,155)-(245,190),
    1,BF
190 PSET (17,160),1
200 PRINT #1,"Barajando."
210 COLOR 1
220 PSET (100,2),10
230 PRINT #1,"PUNTUACION:"
240 COLOR 3
250 D1=20:D2=15
260 RESTORE 2170
270 FOR A=1 TO 52
280 READ S
290 N(A)=S
300 READ S$
310 V$(A)=S$
320 NEXT A
330 REM BARAJANDO LA
    BARAJA
340 FOR A=1TO52
350 D=INT(RND(-TIME)*52)+1
360 FOR U=1 TO A
370 IF W(D)=53THEN GOTO
    350
380 NEXT U
390 C(A)=N(D)
400 W(D)=53
420 NEXT A
430 REM BARAJANDO PALOS
440 FOR A=1 TO 52
450 D=INT(RND(-TIME)*52)+1
460 FOR U=1 TO A
470 IF W(D)=54 THEN GOTO
    450
480 NEXT U
490 P$(A)=V$(D)
500 W(D)=54
520 NEXT A
560 LINE (10,155)-(245,190),
    1,BF
600 FOR A=1 TO 52 STEP 2
610 IF P$(A)="C" THEN GOTO
    2210
620 IF P$(A)="D" THEN GOTO
    2310
630 IF P$(A)="P" THEN GOTO
    2390
640 IF P$(A)="T" THEN GOTO
    2490
810 REM REPRESENTACION DE
    CARTA
820 COLOR 1
830 IF C(A)<=10 THEN GOTO
    900
840 IF C(A)=11 THEN VS$="J"
850 IF C(A)=12 THEN VS$="Q"
860 IF C(A)=13 THEN VS$="K"
870 IF C(A)=14 THEN VS$="A"
880 PSET (D1,D2+2),15:PRINT
    #1,VS$
890 GOTO 910
900 PSET (D1,D2+2),15:PRINT
    #1,C(A)
950 D1=D1+32:IF D1=>230
    THEN D1=10:D2=D2+55
2170 DATA 2,C,3,C,4,C,5,C,6,
     C,7,C,8,C,9,C,10,C,11,C,
     12,C,13,C,14,C
2180 DATA 2,D,3,D,4,D,5,D,6,
     D,7,D,8,D,9,D,10,D,11,D,
     12,D,13,D,14,D
2190 DATA 2,P,3,P,4,P,5,P,6,
     P,7,P,8,P,9,P,10,P,11,P,
     12,P,13,P,14,P
2200 DATA 2,T,3,T,4,T,5,T,6,
     T,7,T,8,T,9,T,10,T,11,T,
     12,T,13,T,14,T
2210 REM CARTAS
2220 REM CORAZONES
2230 LINE (D1,D2)-(D1+30,
     D2+50),15,BF
2240 CIRCLE (D1+12,D2+20),
     3,8
2250 CIRCLE (D1+17,D2+20),
     3,8
2260 PAINT (D1+12,D2+20),
     8:PAINT(D1+17,D2+20),8
```

```
227Ø LINE (D1+9,
     D2+2Ø)-(D1+15,
     D2+32),8
228Ø LINE (D1+2Ø,
     D2+2Ø)-(D1+15,
     D2+32),8
229Ø PAINT (D1+15,D2+3Ø),8
23ØØ GOTO 62Ø
231Ø REM DIAMANTES
232Ø LINE (D1,D2)-(D1+3Ø,
     D2+5Ø),15,BF
233Ø LINE (D1+15,
     D2+2Ø)-(D1+2Ø,
     D2+25),8
234Ø LINE (D1+2Ø,
     D2+25)-(D1+15,
     D2+3Ø),8
235Ø LINE (D1+15,
     D2+3Ø)-(D1+1Ø,
     D2+25),8
236Ø LINE (D1+1Ø,
     D2+25)-(D1+15,
     D2+2Ø),8
237Ø PAINT (D1+15,D2+25),8
238Ø GOTO 63Ø
239Ø REM PIGS
24ØØ LINE (D1,D2)-(D1+3Ø,
     D2+5Ø),15,BF
241Ø CIRCLE (D1+12,D2+3Ø),
     3,1
242Ø CIRCLE (D1+18,D2+3Ø),
     3,1
243Ø PAINT (D1+12,D2+3Ø),
     1:PAINT(D1+18,D2+3Ø),1
244Ø LINE (D1+9,
     D2+3Ø)-(D1+15,
     D2+2Ø),1
245Ø LINE (D1+21,
     D2+3Ø)-(D1+15,
     D2+2Ø),1
246Ø PAINT (D1+15,D2+25),1
247Ø LINE (D1+15,
     D2+3Ø)-(D1+15,
     D2+36),1
248Ø GOTO 64Ø
249Ø REM TREBOLES
25ØØ LINE (D1,D2)-(D1+3Ø,
     D2+5Ø),15,BF
251Ø CIRCLE (D1+15,D2+2Ø),
     3,1
252Ø CIRCLE (D1+12,D2+26),
     3,1
253Ø CIRCLE (D1+18,D2+26),
     3,1
254Ø PAINT  (D1+15,D2+2Ø),1:
     PAINT(D1+12,D2+26),1:
     PAINT(D1+18,D2+26),1
255Ø LINE (D1+15,D2+26)-
     (D1+15,D2+34),1
256Ø GOTO 65Ø
```



La forma de trabajo del programa es la siguiente:

La línea 20 dimensiona cinco variables: C encargada de almacenar las cartas barajadas, p$ encargada de almacenar los palos de las cartas, la N y V$ realizan la misma función que C y P$ pero sólo cuando la baraja esté ya ordenada y W es un simple contador encargado de que no existan dos cartas iguales. La línea 30 nos abre un fichero con el que podemos escribir caracteres en una pantalla en modo de gráficos. La línea 50 elimina las teclas de función y borra la pantalla. La 130 prepara el color, mientras la 150 prepara la pantalla en modo de gráficos de alta resolución. Hasta 240 se diseña el aspecto que tendrá la pantalla durante el juego donde aparecerá el mensaje BARAJANDO. La línea 250 define dos variables D1 y D2 encargadas de seleccionar el lugar de la pantalla donde aparecerá cada carta. En la 260 se indica a qué línea deben referirse las sentencias READ para leer los DATAS. Desde la 270 hasta la 320 se ordena una baraja completa de 52 cartas, de arriba abajo por corazones, diamantes, picas y tréboles. Desde la 330 hasta la 420 se baraja el mazo sin considerar los palos, es decir, en una matriz de 52 se reparten cuatro ases, cuatro dieces y así sucesivamente hasta completar la baraja. La variable W se encarga de que no coincidan dos cartas en la misma posición de la baraja y de que no haya dos iguales. En la línea 390 la carta barajada se archiva en la matriz C. Desde 430 a 520 se barajan sólo los palos, el funcionamiento de esta rutina es igual a la anterior, el palo barajado se archiva en la matriz p$. En 560 se traza un rectángulo de color negro que borra la palabra BARAJANDO que se encuentra en pantalla. La 600 crea un bucle con cabida para 26 peticiones de carta. Desde 610 a 640 se comprueba qué palo es la última carta pedida y se manda el control a la rutina que dibuja la carta correspondiente. Desde la línea 810 hasta la 910 se programa la representación del símbolo de la carta encima de su dibujo, si el símbolo de la carta está entre dos y diez es representado por la línea 900, si se encuentra entre once y catorce será sustituido por los símbolos J, Q, K o A y representado en la línea 880. La línea 950 se encarga de desplazar la posición de la pantalla donde se representará la siguiente carta. Entre 2170 y 2200 se define una baraja, desde 2210 hasta 2560 se programan cuatro rutinas que dibujan los distintos palos.



EMPIEZA EL JUEGO

- REPARTIENDO LAS CARTAS
- LAS APUESTAS
- DOBLAR, CAMBIAR Y PLANTARSE
- HACER SALTAR LA BANCA
- EJECUCION DE LOS TOTALES

El banquero fija en ti su mirada de hielo. Haces tu apuesta y recibes otra carta, pero ¿pides carta o te plantas? En esta ocasión nos ocuparemos de las líneas de programa correspondientes a la parte del jugador.

Continuando a partir de la rutina de gráficos que tecleaste en el capítulo anterior, ahora te hacen falta dos secciones más de programa, una para manejar las respuestas del jugador y otra para hacer posible que juegue el ordenador. En este capítulo veremos las líneas que hacen falta para el jugador y para las fichas, pero no llegaremos muy lejos jugando ahora, ya que todavía no has enseñado a jugar a tu ordenador.

Esta sección del programa está relacionada con unas cuantas tareas diferentes. No te preocupes si no estás muy seguro de las reglas exactas del juego de las veintiuna, ya nos ocuparemos de ellas junto con la última parte del programa. Básicamente el programa tiene que hacer tres cosas, ocuparse del reparto de las cartas, permitir al jugador hacer apuestas, y pedir cartas adicionales y finalmente tiene que calcular la puntuación del jugador.

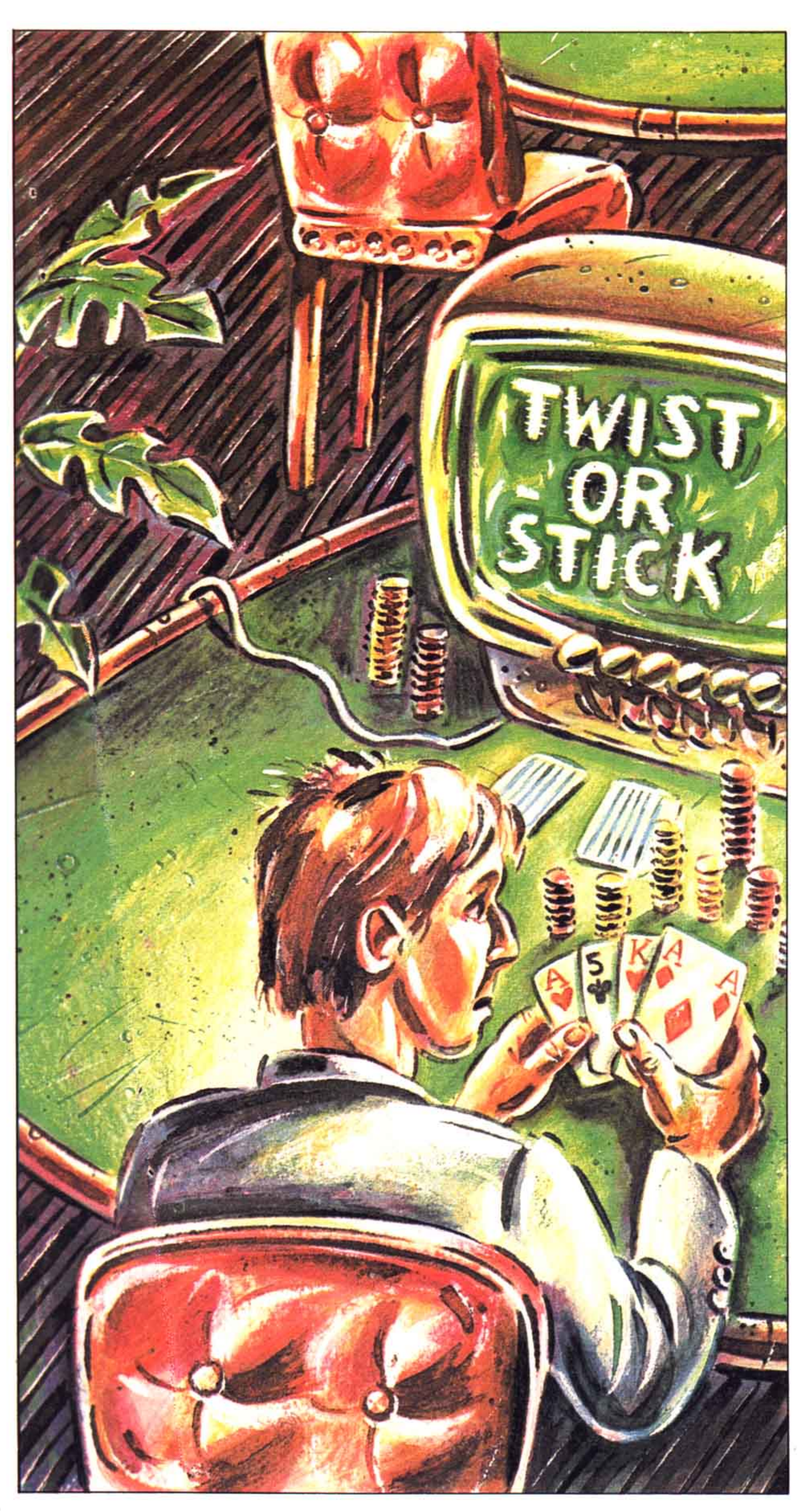
La siguiente sección del programa contabiliza los puntos del jugador, comprueba que no se haya pasado de veintiuno, o que sea igual a veintiuno o que esté entre veintiuno y dieciséis para ofrecer la posibilidad de plantarse. Después el ordenador pide al jugador que haga su apuesta por esta carta; tras hacerlo el ordenador muestra la cantidad de dinero apostado sobre la mesa, la cantidad que le queda al jugador y la que le queda a la banca. El programa además presenta mensajes al jugador dependiendo de la jugada, como: has hecho 21, te ha salido un as, etc...



Agrega ya las siguientes líneas al programa del capítulo anterior.

```
65Ø FOR B=2 TO 1Ø
66Ø IF C(A)=B THEN PA=
    PA+B
67Ø NEXT B
71Ø IF C(A)=11 THEN
    PA=PA+1Ø
72Ø IF C(A)=12 THEN
    PA=PA+1Ø
73Ø IF C(A)=13 THEN
    PA=PA+1Ø
78Ø IF C(A)=14 THEN GOTO
    2Ø8Ø
91Ø LINE (2ØØ,Ø)-(255,1Ø),1Ø,
    BF
92Ø PSET (2ØØ,2),1Ø
93Ø PRINT #1,PA
94Ø COLOR 3
96Ø IF PA>21 THEN GOTO
    124Ø
98Ø IF PA=21 THEN GOTO
    144Ø
1ØØØ IF PA=>16 AND PA<21
     THEN 164Ø
1Ø2Ø LINE (1Ø,155)-(245,
     19Ø),1,BF
1Ø3Ø PSET (17,16Ø),1
1Ø4Ø PRINT #1,
     "APUESTA."
1Ø5Ø PSET (17,17Ø),1
1Ø6Ø PRINT #1,"1-1ØØ$,
     3-1ØØØ$"
1Ø7Ø PSET (17,18Ø),1
1Ø8Ø PRINT #1,"2-5ØØ$,
     4-1ØØØØ$"
1Ø9Ø J$=INKEY$
11ØØ IF J$="1" THEN
     JA=JA+1ØØ:
     GOTO115Ø
111Ø IF J$="2" THEN
     JA=JA+5ØØ:
     GOTO115Ø
112Ø IF J$="3" THEN
     JA=JA+1ØØØ:
     GOTO115Ø
113Ø IF J$="4" THEN
     JA=JA+1ØØØØ:
     GOTO115Ø
114Ø GOTO 1Ø9Ø
115Ø LINE (15,13Ø)-(23Ø,
     148),1,BF
116Ø PSET (15,14Ø),12
117Ø PRINT #1,JA*2;"$H";
     AP-JA;"$M-";
     PP-JA
118Ø LINE (1Ø,155)-(245,
     19Ø),1,BF
119Ø PSET (17,16Ø),1
12ØØ PRINT #1,"CUBRO LA
     APUESTA."
121Ø FOR WE=1 TO 5ØØ:NEXT
     WE
122Ø NEXT A
124Ø LINE (1Ø,155)-(245,
     19Ø),1,BF
125Ø PSET (17,16Ø),1
126Ø PRINT #1,"TE HAS
     PASADO,HAS HECHO:";
     PA
127Ø PSET (17,17Ø),1
128Ø PRINT #1,"YO GANO."
129Ø FOR WE=1 TO 5ØØ:NEXT
     WE
13ØØ PP=PP+JA:AP=
     AP-JA
131Ø IF PP=<Ø THEN GOTO
     257Ø
132Ø GOTO 178Ø
144Ø LINE (1Ø,155)-(245,
     19Ø),1,BF
145Ø PSET (17,16Ø),1
146Ø PRINT #1,"HAS HECHO
     21."
147Ø PSET (17,17Ø),1
148Ø PRINT #1,"TU GANAS."
149Ø FOR WE=1 TO 5ØØ:NEXT
     WE
15ØØ AP=AP+JA:PP=PP-JA
151Ø IF PP=<Ø THEN GOTO
     257Ø
152Ø GOTO 178Ø
164Ø LINE (1Ø,155)-(245,
     19Ø),1,BF
165Ø PSET (17,16Ø),1
166Ø PRINT #1,
```






```
      "(1)PLANTARSE."
1670 PSET (17,170),1
1680 PRINT #1,"(2)PEDIR
      CARTA."
1690 K$=INKEY$
1700 IF K$="1" THEN GOTO
      1880
1710 IF K$="2" THEN GOTO
      1010
1720 GOTO 1690
1780 LINE (10,155)-(245,
      190),1,BF
1790 PSET (17,160),1
1800 PRINT #1,"(1)VOLVER A
      JUGAR."
1810 PSET (17,170),1
1820 PRINT #1,
      "(2)RETIRARTE."
1830 K$=INKEY$
1840 IF K$="1" THEN GOTO
      130
1850 IF K$="2" THEN END
1860 GOTO 1830
1880 LINE (10,155)-(245,
      190),1,BF
1890 PSET (17,160),1
1900 PRINT #1,"TE PLANTAS
      Y"
1910 IF PA>PB THEN PSET (17,
      170),1:PRINT #1,
      "GANAS."; PA;"a";
      PB:FOR WE=1 TO
      500:NEXT WE:AP=AP+
      JA:PP=PP-JA:GOTO 1950
1920 IF PB>PA THEN PSET (17,
      170),1:PRINT #1,
      "PIERDES."; PB; "a";
      PA:FOR WE=1 TO
      500:NEXT WE:AP=AP-
      JA:PP=PP+JA:GOTO 1950
1930 FOR WE=1 TO 500:NEXT
      WE
1940 PSET (17,170),1:PRINT
      #1,"HEMOS
      EMPATADO.":FOR WE=1
      TO 500:NEXT WE
1950 IF PP=<0 THEN GOTO
      2570
1960 GOTO 1780
2080 LINE (10,155)-(245,
      190),1,BF
2090 PSET (17,160),1
```

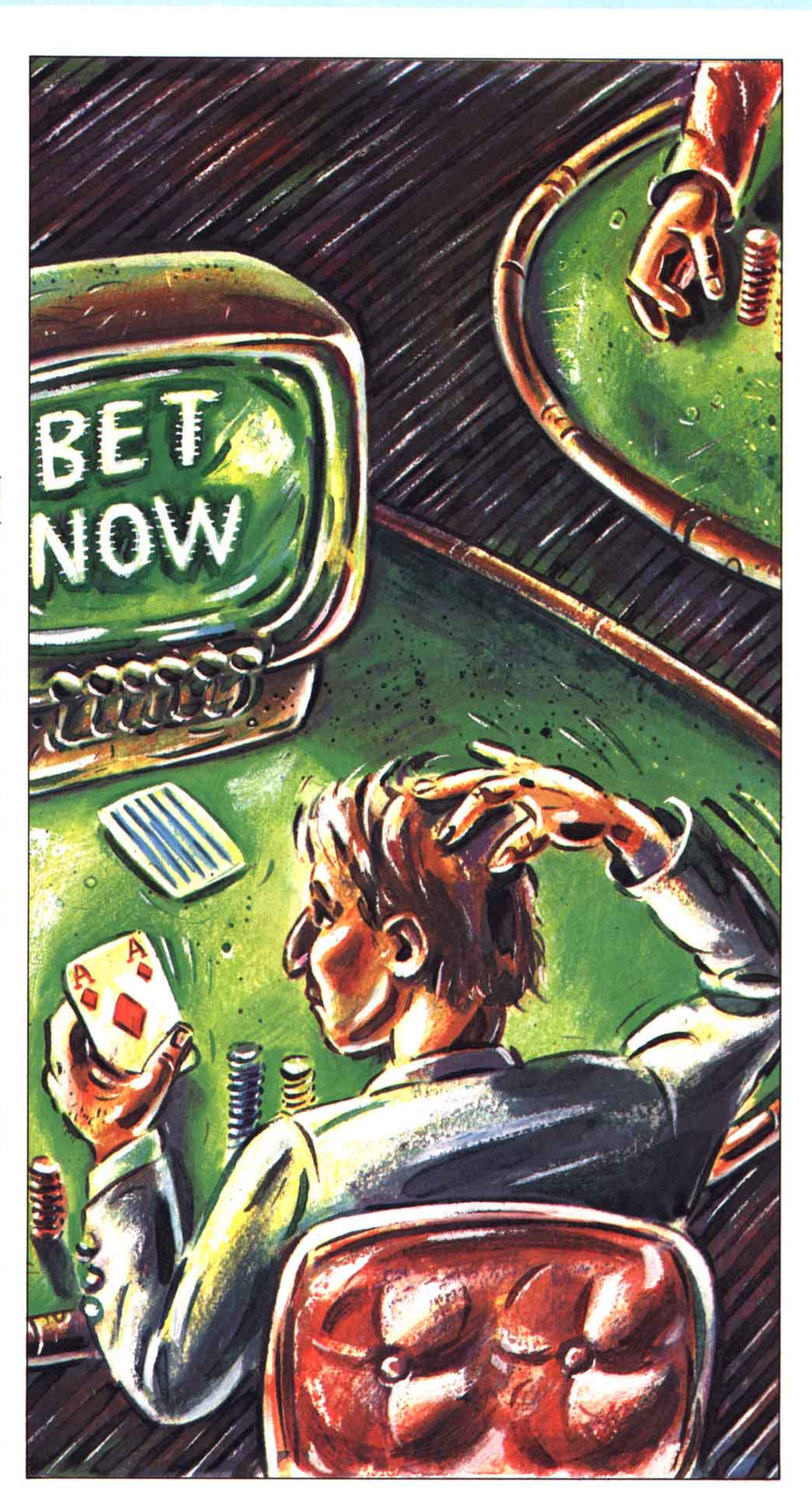
```
2100 PRINT #1,"TE HA SALIDO
     UN AS"
2110 PSET (17,170),1
2120 PRINT #1,"(1)SUMAR
     1-(2)SUMAR 11"
2130 K$=INKEY$
2140 IF K$="1" OR K$="2"
     THEN GOTO 2150 ELSE
     2130
2150 IF K$="1" THEN
     PA=PA+1 ELSE
     PA=PA+11
2160 GOTO 790
```

MÁS CARTAS

Las líneas 650 a 670 comprueban si la carta del jugador no es una figura, en cuyo caso la variable PA es sumada al valor original de la carta. Desde 710 a 730 se comprueba si la carta es una figura, sumando entonces 10 a PA. La línea 780 comprueba si es un as, en cuyo caso manda el control a una rutina que nos pregunta si queremos sumar 1 u 11. Desde la línea 910 a 940 se imprime la puntuación total en la pantalla y la 960 se cerciora de que no nos hayamos pasado de 21. De haberlo hecho el ordenador contestaría: TE HAS PASADO, YO GANO. La línea 980 comprueba si hemos hecho 21, en cuyo caso nos diría HAS HECHO 21, TU GANAS; en caso de que nuestros puntos superen 16 la línea 1000 pasa el control a una rutina que nos pregunta si queremos plantarnos o pedir carta.

APUESTAS

Entre 1020 y 1140 se extiende una rutina que nos pregunta la cantidad que queremos apostar, esta cantidad es guardada carta tras carta por la variable JA, en 1170 se nos muestra en pantalla la cantidad de dinero apostado sobre la mesa, la cantidad en disposición del jugador y la cantidad en disposición de la máquina, el resto de esta parte del programa está formado por subrutinas de mensajes como, por ejemplo, volver a jugar o retirarte, te plantas y ganas, te plantas y pierdes.



HACER SALTAR LA BANCA (I)

- EL ORDENADOR Y EL JUEGO DE CARTAS
- JUGANDO CON EL ORDENADOR
- COMO FUNCIONA EL PROGRAMA
- LAS PIEDRAS

En esta oportunidad trataremos de convertir en realidad el sueño que calladamente acaricia todo jugador: hacer saltar la banca.

La cantidad de dinero que le resta a la máquina es igual a la variable PP, que almacena todo su dinero, menos JA que es igual a la cantidad apostada. Si la cantidad de dinero que le resta a la banca es igual a cero, será detectado en las rutinas de mensajes y se enviará el control a la línea 2570 donde aparecerá un mensaje de felicitación y habremos ganado; el control retorna entonces al principio, preguntándonos de cuánto dinero disponemos.

Para completar el juego de las veintiuna tienes que programar tu ordenador para que juegue cuando le toque el turno. Aquí tienes, además, una descripción de las reglas del juego para el caso de que no estés familiarizado con el mismo.

LAS REGLAS DEL JUEGO

Antes de pasar a examinar el resto de la programación, vale la pena recapitular las reglas del juego.

El juego de las veintiuna se juega con un mazo ordinario de 52 cartas. Las cartas del 2 al 10 valen lo que indica su número; las figuras valen 10, y el as vale 1 u 11 según las necesidades del jugador. En este juego el ordenador se ocupa de ir llevando la cuenta de los totales, por lo que no tienes que hacerlo tú.

Normalmente se juega con dinero, o con «piedras», pero en esta versión para ordenador hay que programarle para que cuente la puntuación en fichas imaginarias; con las otras fichas un programador poco escrupuloso podría hacer que la máquina dejara de ser honrada.

El ordenador se programa para actuar como banquero en todo momento y será siempre el encargado de repartir las cartas.

Al principio del juego se barajan las cartas y se sacan dos de ellas poniéndolas boca abajo. En la pantalla aparecerá boca arriba la carta del jugador, aunque el programa ha sido diseñado para que la máquina no sepa qué carta es la que tiene el jugador.

El jugador debe apostar ahora sobre esta primera carta, antes de que se saque una nueva para cada uno.

El objeto del juego es terminar con mejor puntuación que la banca, es decir con un valor total más elevado. Una mano que sume en total más de 21 se pasa y pierde. Una mano con puntuación entre 16 y 21 solamente vence a la banca si la máquina tiene una mano más baja o se ha pasado de 21.

Las veintiuna del jugador vencen a todo lo que tenga la máquina excepto a otras veintiuna.

EL PROGRAMA

En primer lugar el ordenador se presenta y nos pregunta de qué cantidad de dinero disponemos. Tras esto baraja las cartas y reparte una al jugador y otra para él mismo, nos indica la puntuación total hasta el momento y nos pide que hagamos una apuesta por esa carta. En la parte inferior de la pantalla, aparece un menú con opciones para apostar entre 100 y 10.000 dólares. Presionando el número seguido de un guión que aparece al lado de cada cifra elegiremos la apuesta. Una vez hecho esto, el programa nos mostrará la cantidad de dinero apostado que se encuentra sobre la mesa, el dinero que nos queda y la cantidad que le resta a la máquina.



CODIGO MAQUINA AL ALCANCE DE TODOS

- CODIGO MAQUINA
- INTRODUCCION AL LENGUAJE ENSAMBLADOR
- MNEMONICOS
- REGISTROS

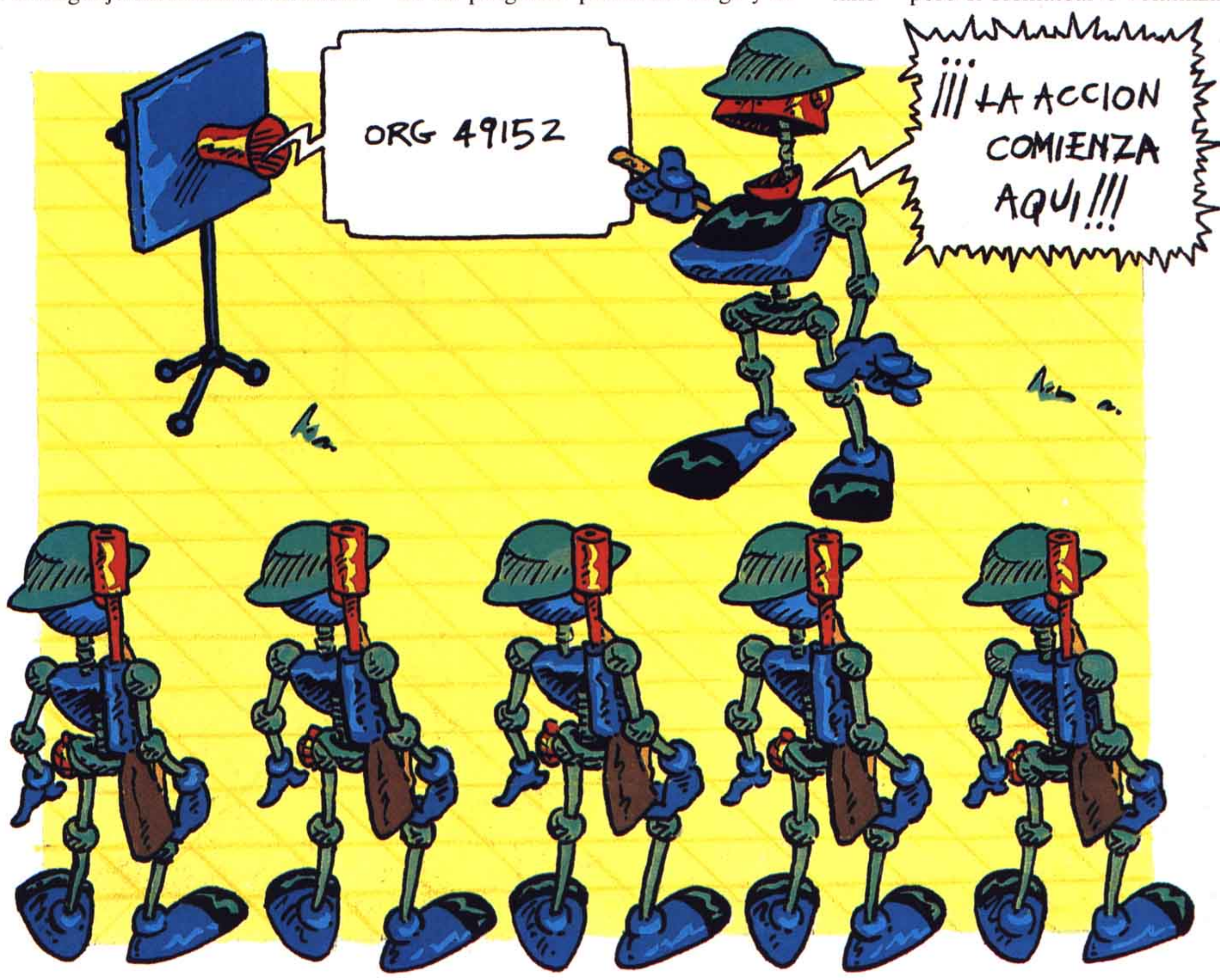
A partir de este número damos comienzo a una nueva sección dedicada a aprender lenguaje ensamblador de un modo sencillo. Para facilitar la comprensión combinaremos explicaciones teóricas con ejemplos prácticos. Al mismo tiempo, iremos confeccionando un apasionante juego.

Las diferencias entre los vocablos lenguaje binario, lenguaje hexadecimal (código máquina) y lenguaje ensamblador, se verán en la primera parte de la sección. ¿Para qué se utiliza el lenguaje ensamblador en MSX?

El lenguaje ensamblador es un lenguaje próximo al que habla el ordenador, utilizándose cada vez que una aplicación (o juego) es irrealizable desde el BASIC o si es realizable la velocidad de ejecución es muy lenta. Su perfecto conocimiento permitirá al usuario ahorrar memoria aprovechando al máximo las posibilidades de su ordenador.

Los programas escritos en lenguaje ensamblador son de ejecución rápida pero extraordinariamente difíciles de escribir y depurar. La puesta a punto de un programa puede ser larga y laboriosa. Así como en BASIC un simple error de tecleado o de estructura produce un mensaje del tipo «Type Mismatch» o «Syntax Error in...», en lenguaje ensamblador se corre el peligro de bloquear el ordenador completamente, lo que significa en la mayoría de los casos *resetear* éste y perder en su totalidad lo almacenado en memoria.

No obstante, un programa mal escrito en ensamblador no puede jamás destruir un ordenador —los constructores ya han tenido en cuenta ese detalle— pero sí formatear o volatilizar



el contenido del diskette si éste no está protegido contra escritura («Write Protect»). Es por ello imprescindible tener siempre a mano una copia de seguridad para evitar la pérdida de tiempo que supone teclear de nuevo el programa.

INTRODUCCIÓN AL LENGUAJE ENSAMBLADOR

A continuación explicaremos unas nociones necesarias para comprender el lenguaje ensamblador. Podrán parecer un tanto repetitivas, o demasiado fáciles para los ya *iniciados* en la *verdadera programación*, pero siempre es deseable y útil el repasarlas.

Lenguaje máquina o código objeto

Al nivel más elemental un ordenador, sea cual sea, sólo comprende impulsos eléctricos. Las instrucciones que se dan al microprocesador (el corazón del ordenador) son una sucesión, más o menos larga, de presencias y ausencias de impulsos eléctricos, algo parecido a lo que sucede en lenguaje *morse*.

El lenguaje (código) máquina está constituido por una serie de órdenes determinadas que se traducen cada una por una sucesión determinada de impulsos eléctricos. Con el fin de representar las diferentes sucesiones de impulsos eléctricos, los creadores decidieron utilizar las cifras 0 y 1. Un impulso eléctrico se traduce por la notación 1 y su ausencia por la notación 0. Esta forma de transcribir el código máquina a unos y ceros se denomina *lenguaje binario*.

Sin embargo, este código es difícil de memorizar puesto que cada serie de impulsos representan una instrucción compuesta de ocho elementos (ocho estados de ausencia/presencia, de 0 y 1).

¿Por qué ocho elementos?

El ordenador MSX está construido alrededor de un *microprocesador Z-80* capaz de tratar ocho impulsos simultáneamente. Por eso se dice que el Z-80 es un microprocesador de ocho bits. Sobre este tema os será de mucha utilidad la consulta de nuestra sección *El Micro Z-80*.

Para más facilidad, estos bloques de ocho bits, de ocho impulsos, se transcriben en forma **hexadecimal**. Cada bloque de ocho bits se escinde en dos bloques de cuatro bits *(nibbles)* representados cada uno por un número comprendido entre 0 y F: 0,1,...,9,A, ...F

El cuadro 1 nos muestra los diferentes valores que puede tomar un bloque de 4 bits (cuadro 1).

Así, la serie de ocho bits:

10101110

se escribirá:

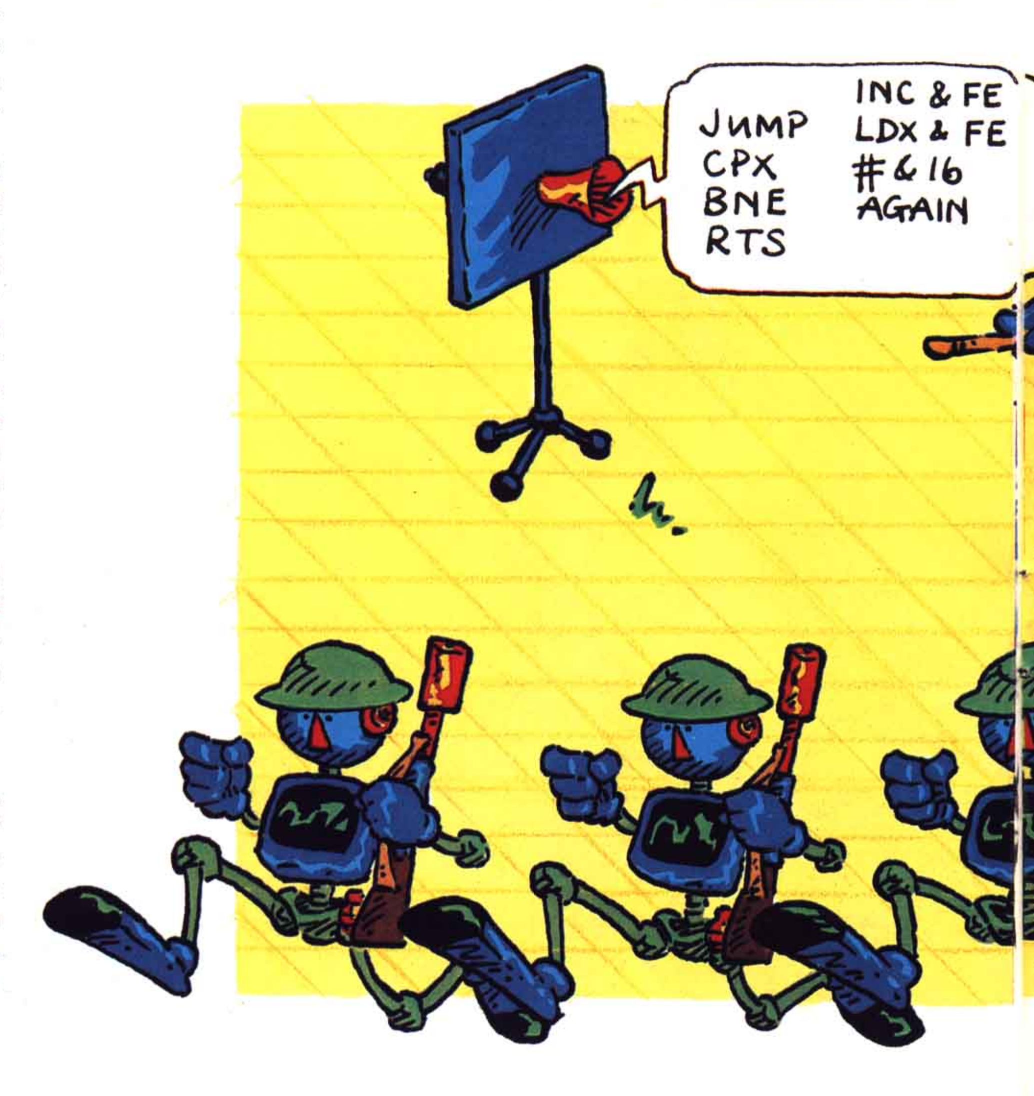
A E

que es una notación más concisa, y por consecuencia, más fácil de retener. Esta notación hexadecimal es generalmente llamada lenguaje (código) máquina (cuadro 2).

Mnemónicos y lenguaje ensamblador

Para representar las diferentes instrucciones y operaciones del microprocesador, el lenguaje máquina es suficiente pero a su vez, de nuevo poco práctico para su uso. Incluso para un ojo experimentado pueden aparecer los programas como una sucesión de números desprovista de significado, debido a que las instrucciones, los datos y las direcciones aparecerán como cadenas de dígitos hexadecimales unidos de uno a otro extremo.

La etapa siguiente consistirá en representar las instrucciones del microprocesador por medio de abreviaturas extraídas de palabras inglesas que representan la función efectuada. Es decir, las abreviaturas que representan los códigos de operación de la máquina son prácticamente autoexplicativas, volvemos a decir que en inglés, y que veremos seguidamente. Los da-



tos y las direcciones son números exactamente iguales que los que figuran en el código máquina.

A las instrucciones así designadas se les da el nombre de **mnemónicos** (de la raíz griega que significa *memoria* en el sentido de *fácil de recordar*).

Con los códigos de operación escritos en forma de **mnemónico**, la secuencia ininterrumpida de números en código máquina queda fraccionada de forma que se puede ver lo que está sucediendo. Así en el ejemplo siguiente veremos, en la columna de la izquierda, una suma en lenguaje hexadecimal, y en la de la derecha su equivalente en lenguaje ensamblador.

```
3505        LDA,5
3503        LDB,3
 80       ADD.A,B
```

Cuando se está ensamblando en código máquina, lo único que se tiene que hacer es consultar los **mnemónicos** apropiados en la correspondiente tabla que figura en la guía del microprocesador que lleva el ordenador. Así, el MSX, lleva un *microprocesador Z-80*, como vimos anteriormente. La tabla dará el código numérico de operación. Sólo debemos acordarnos de permutar entre sí los bytes alto y bajo de todas las direcciones y datos que vengan en grupos de dos bytes. Esto se debe a que el ordenador almacena los números en formato byte bajo/byte alto. Aquí vemos cómo se introduciría en un registro, por ejemplo el LDHL, el contenido &H4000. Expresado en lenguaje máquina se convertiría en 21-00-40. Debemos evitar el error de escribirlo en el mismo orden que en el ensamblador.

Por el término ensamblador se entienden dos conceptos: el primero designa el **lenguaje** ensamblador, el mismo que hemos descrito con anterioridad. El segundo designa a un **programa**, escrito asimismo en lenguaje ensamblador, que permite transformar un programa escrito con la ayuda de **mnemónicos** (programa FUENTE) en un programa en lenguaje máquina (programa OBJETO).

A continuación vamos a hablar de unos cuantos **mnemónicos** de lenguaje ensamblador para el microprocesador del MSX. Veamos el significado de algunos de ellos:

LD significa *cargar* (LOAD); **J** significa *saltar* (JUMP); **ADD**, *sumar*; **CALL**, *llamar a una subrutina*; **INC B**, *incrementar, añadir 1, el registro B*; **RES**, *poner un bit a cero* (resetear); **JRNZ**, *salto relativo si el resultado no es cero*; **SET**, *poner un bit a uno;...*

Como podemos ver, estos **mnemónicos**, igual que los códigos hexadecimales a los que representan, sirven para establecer el valor de un indicador y saltar hacia una u otra parte del programa. Éstas son las únicas cosas que pueden hacer las instrucciones en código máquina.

La traducción de lenguaje ensamblador a código máquina no es demasiado fácil. Si observamos el conjunto de instrucciones en la guía del microprocesador, encontraremos que una sencilla instrucción como **LDA** (cargar el registro A o acumulador) puede tener varios códigos de operación distintos (en este caso 16). Por ello debemos decidir cuál de estos códigos de operación es el que nos interesa. El número total de **mnemónicos** es de 707.

Los diferentes códigos de operación dependen del tipo de direccionamiento que se utilice. Se llaman modos de direccionamiento a las diferentes maneras en que el microprocesador accede a la información.

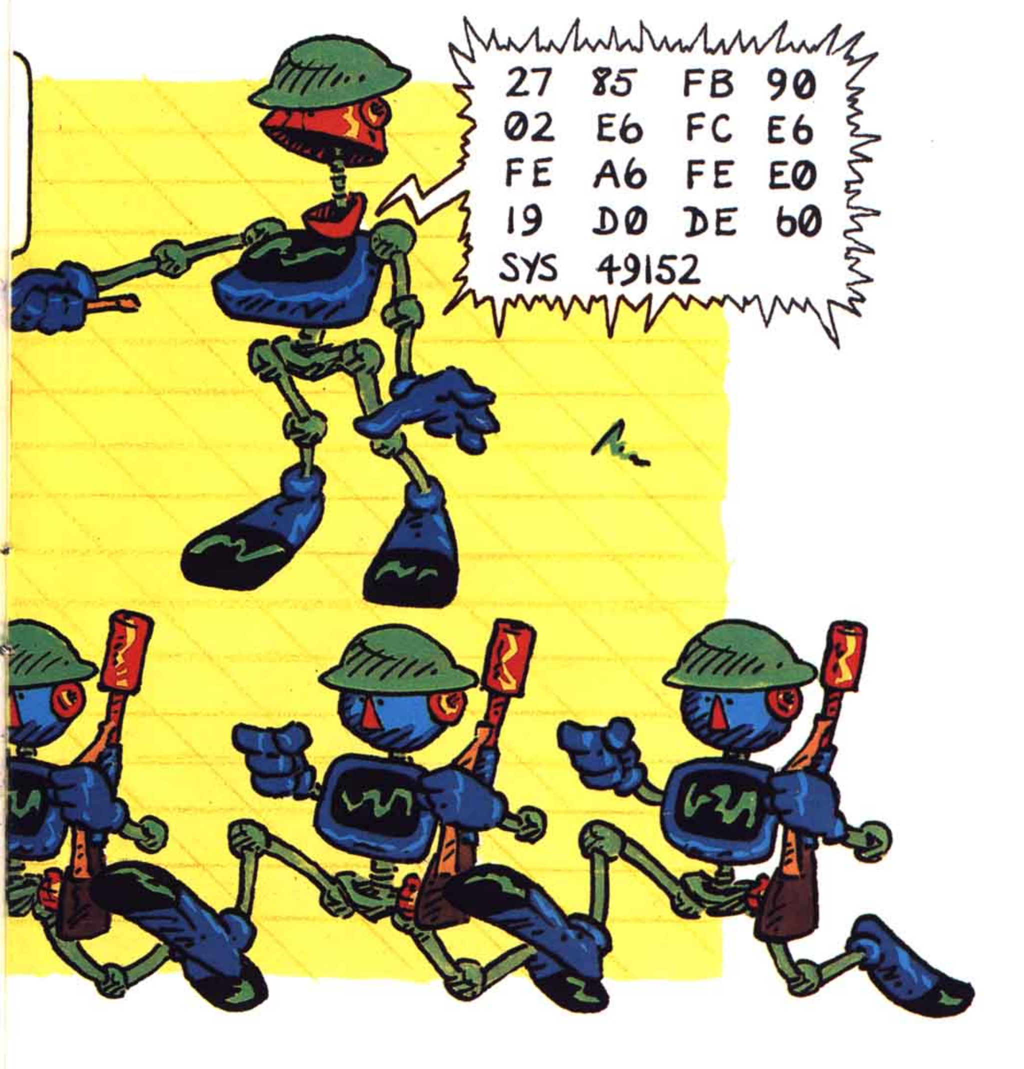
El modo de direccionamiento más sencillo del Z-80 es el direccionamiento inmediato. Se expresa mediante un número que sigue inmediatamente al **mnemónico** del lenguaje ensamblador.

Por ejemplo:

LD A,4

;cargar el registro A con el número 4

;en hexadecimal sería 3E 04



CUADRO 1

Decimal	Binario	Hexadecimal
Ø	ØØØØ	Ø
1	ØØØ1	1
2	ØØ1Ø	2
3	ØØ11	3
4	Ø1ØØ	4
5	Ø1Ø1	5
6	Ø11Ø	6
7	Ø111	7
8	1ØØØ	8
9	1ØØ1	9
1Ø	1Ø1Ø	A
11	1Ø11	B
12	11ØØ	C
13	11Ø1	D
14	111Ø	E
15	1111	F

LD A,(nn)
;cargar el contenido de la dirección de
;memoria (nn) en el registro A
;3A (bytes de dirección de memoria permutados).

El direccionamiento directo también funciona en sentido contrario:

LD (nn),A
;cargar el contenido del registro A en la
;dirección de memoria (nn);32 (nn).

Si se utiliza el direccionamiento indirecto, le decimos a la máquina dónde puede encontrar la dirección del dato que necesitamos:

LD A,(HL)
;cargar el registro A con el dato contenido
;en la dirección que hay almacenada en el
;registro HL.

LD (HL),A
;cargar el contenido del registro A en la
;posición de memoria cuya dirección está
;almacenada en el registro HL.

Existe un tipo especial de direccionamiento indirecto que se llama direccionamiento indexado. En este caso se utiliza uno de los dos registros de índice —IX e IY— y la dirección real que hay que utilizar en la operación se obtiene mediante un desplazamiento que se suma al contenido del registro de índice X o Y:

LD A,(IX+d)

Pueden transferirse datos de un registro a otro (direccionamiento de registro a registro):

LD D,B
;cargar el registro D con el contenido del
;registro B.

El direccionamiento relativo sólo se utiliza con las instrucciones de salto que son equivalentes a las instrucciones GOTO del BASIC; con ellas se le dice al ordenador cuántos bytes tiene que saltar hacia adelante o hacia atrás:

JRNZ,FC
;en hexadecimal 20 FC.

Este direccionamiento relativo se utiliza muy poco en el lenguaje ensamblador. Lo normal es que los saltos se indiquen por medio de *etiquetas*. Se trata de palabras utilizadas como marcas para señalar el principio y el fin de los bucles.

La marca se pone delante de la instrucción a la que hay que saltar y después de la instrucción de salto:

BUCLE LD A,n

y posteriormente:

DJNZ BUCLE
;si el indicador de cero no está activado decrementar en 1 el registro B y
;saltar al lugar donde esté la etiqueta
;*bucle* delante de una instrucción.

Con esto finalizamos la primera parte dedicada al código máquina en MSX. Queremos recomendaros que leáis el artículo sobre la *Estructura y arquitectura del Z-80* que publicamos en este mismo número. En esta sección de Código Máquina continuaremos hablando de las variables internas del sistema, las rutinas del BIOS, y pequeños programas en lenguaje ensamblador, que nos permitirán adentrarnos en el fascinante mundo de la programación.

También comenzaremos a programar un magnífico juego.

CUADRO 2

Para pasar un número de notación decimal a hexadecimal, podemos utilizar:

PRINT HEX$ (número decimal)

Ejemplo: queremos pasar a hexadecimal el número decimal 32768:

PRINT HEX$ (32768)<RETURN>
8ØØØ[=&H8ØØØ]

Si queremos hacerlo a la inversa, haremos en el ejemplo anterior:

n.° decimal=16.^{0}*(primer díg por la der.) + 16.^{1}*(2.° dígito por la der.) + 16.^{2}*(3.er dígito por la der.) + 16.^{3}*(4.° dígito por la der.)

siendo la fórmula general de conversión de hexadecimal a decimal.

!PARTICIPA EN EL CONCURSO!

En INPUT estamos convencidos de que aún puedes hacer muchas más cosas con tu ordenador. Sin duda, muchos lectores estareis utilizando vuestro micro para funciones de lo más variadas, en unos casos; pintorescas, en otros; mientras que algunos listillos habrán podido utilizarlo para resolver tareas complejas. Es lógico, modificando programas y variando los periféricos nuestro ordenador puede prestar sus servicios en infinidad de facetas. INPUT quiere que esas aplicaciones y utilidades a las que has conseguido dedicar tu ordenador, sean conocidas por todos sus lectores y por eso ha organizado el «Concurso de Aplicaciones y Utilidades», en el que puede participar cualquiera de nuestros lectores.





UTILIDADES Y APLICACIONES: Si tu ordenador controla la calefacción de tu casa, gobierna un robot, dirige un pequeño negocio, organiza la maqueta de tu tren eléctrico, o cualquier cosa interesante u original; envíanos información gráfica y listados de tus programas, grabados en un cassette, diskette o microdrive.

Todo ello habrá de venir acompañado por un texto que aclare cuál es su objetivo, el modo de funcionamiento y una explicación del cometido que cumplen las distintas rutinas que lo componen. El texto se presentará en papel de tamaño folio y mecanografiado a dos espacios. No importa que la redacción no sea muy clara y cuidada; nuestro equipo de expertos se encargará de proporcionarle la forma más atractiva posible.

UN JURADO propio decidirá en cada momento qué colaboraciones reúnen los requisitos adecuados para su publicación, y evaluará la cuantía del premio en metálico al que se hagan acreedoras.

No olvideis indicar claramente para qué ordenador está preparado el material, así como vuestro nombre y dirección y, cuando sea posible, un teléfono de contacto. Entre todos los trabajos recibidos durante los próximos tres meses SORTEAREMOS:

- **Un premio de 50.000 ptas.**
- **Un premio de 25.000 ptas.**
- **Un premio de 10.000 ptas.**
  en material microinformático a elegir por los afortunados.

¡No os desanimeis!, por muy simples o complejas que puedan parecer vuestras ideas, todas están revisadas con el máximo interés.

INPUT MSX
Aribau, 185. Planta 1.ª
08021 BARCELONA

NOTA: INPUT no se responsabiliza de la devolución del material que no vaya acompañado por un sobre adecuado con el franqueo correspondiente.

COMPOSICIÓN DE ETIQUETAS

Este sencillo programa te va a permitir componer el texto de las etiquetas de tus microfloppys.

A primera vista quizá no te parezca demasiado interesante, ya que rotularlas a mano no te cuesta ningún esfuerzo; pues bien, recuerda cuántas veces has malgastado etiquetas porque no tenías muy claro lo que querías escribir, o porque sencillamente no te cabía.

Además de estas razones hay otro concepto a tener presente, como es el orden, argumento de peso en el acabado de los trabajos.

Está claro que el tener una etiqueta bien formateada, no contribuirá a que un programa funcione mejor, pero sí va a dar un toque final a la presentación del trabajo, además de simplificar la confección de la etiqueta.

Al ejecutar el programa veremos en la pantalla el formato gráfico de una etiqueta, y un cursor que podemos desplazar en todas direcciones mediante las teclas de *movimiento de cursor*.

Con dicho cursor podemos situarnos en cualquier línea de la etiqueta, no siendo necesario hacerlo ordenadamente.

Una vez compuesto se trata de imprimir el texto en la impresora, donde previamente habremos colocado una etiqueta virgen.

El programa carece de presentación, dado que el planteamiento del mismo no es el de un juego o entretenimiento, sino que lo hemos considerado como un útil de trabajo.

Su funcionamiento nos permitirá ver la etiqueta con el texto, antes de que exista realmente.

DESCRIPCIÓN DEL PROGRAMA

El primer paso es dibujar en la pantalla una etiqueta con las características y proporciones reales. Este dibujo queda de la siguiente forma: fondo de la pantalla de color gris, etiqueta blanca y tinta de color azul.

Para este desarrollo trabajamos con la SCREEN 6.

Hemos definido los colores, alterando la paleta de colores de la máquina, y adjudicándoles un valor numérico que será al que haremos referencia durante el programa.

Estos serán: 0 para el negro, 1 para el azul, 2 para el gris y 3 para el blanco, que localizamos en las líneas 250 a 280.

Los comandos de la línea 290 definen cómo es el papel, borde y tinta.

Otra información para el microordenador es definir en qué lugar de la pantalla debe dibujar, para ello se crean unas coordenadas, la *x* en la línea 320 y la *y* en la línea 330.

Una vez dimensionados el ancho, interlineado (espacio entre líneas), anchura total, altura total, y coordenadas de inicio, se dibuja un cuadrado lleno de color blanco con el borde de color azul, mediante la instrucción LINE que encontramos en las líneas 420 y 430. En la 450 entramos en un bucle encargado de dibujar las seis líneas donde va el texto, hasta llegar a la línea 480.

En la línea 490, llenamos de color azul el pie o parte baja de la etiqueta. Seguidamente nos dirigimos a las líneas 510 a 530, que sitúan el cursor donde debe escribir «INDEX», es decir al margen superior izquierdo, pintando sobre fondo blanco con tinta azul.

En los comandos de las líneas 550 a 590 nos sucede algo muy parecido, aunque exactamente a la inversa, o sea, definimos el fondo azul y pintamos con tinta blanca.

En la línea 560 ponemos el cursor en el origen donde queremos escribir WRITE PROTECT.

En la 580 lo situamos para escribir WRITE ENABLE.

A partir de ese momento, la etiqueta ya está dibujada.

A continuación, inicializamos las seis líneas de texto con el rótulo *Línea de texto disponible*, indicando cuáles son las partes que podemos rellenar. Esto sucede en las líneas 620 a 690.

Para poder situar el texto dentro de la etiqueta ha sido necesario definir dos variables, que forman parte del bucle principal que va de la línea 710 a 880. Dichas variables son: la F para la fila, y la C para la columna, colocando el cursor en la parte superior izquierda de la pantalla con una posición inicial F=1 y C=1.

De aquí pasa a la subrutina que comienza en la línea 1420 encargada de calcular las coordenadas gráficas que corresponden a esa fila y columna, en función de donde esté definida la etiqueta antes mencionada en las líneas 320 y 330. Estas coordenadas *x* e *y*, pueden ser variadas por el usuario.

Cada vez que modificamos un carácter, esta subrutina comprobará su situación en la pantalla gráfica. Su existencia es justificable por una sencilla razón, es necesario calcular el valor de las coordenadas, ya que la etiqueta puede dibujarse en cualquier punto de la pantalla manteniendo siempre, esté donde esté, los valores F=1 y C=1 para el inicio del texto. Estos puntos no tendrían el mismo valor en la pantalla gráfica, debiendo calcular la posición de cada carácter entrado por el teclado.

Otra cuestión sería el movimiento de cursor, para lo cual en la línea 740, y para facilitar la localización del cursor sobre el fondo blanco, escribiremos con tinta azul sobre papel gris, quedando el cursor de color gris.

Seguidamente escribimos el carácter correspondiente sobre el cursor, utilizando una matriz T$ que hemos

- UN PROGRAMA MUY SENCILLO
- ETIQUETAS BIEN FORMATEADAS
- MOVIMIENTO DEL CURSOR
- MAS VENTAJAS
- DESCRIPCIÓN DEL PROGRAMA
- COLOR DE LA TINTA Y EL PAPEL
- CONFECCIÓN DE LA ETIQUETA

formateado en la línea 220, y que consta de seis cadenas de caracteres.

En cada cadena se almacenará una línea de texto.

En los comandos de la línea 750 buscamos cuál es el carácter de la cadena T$ de la fila y columna donde se halla situado el cursor.

La función dice cuál es el carácter que va en la celdilla, fila y columna.

En las líneas 770 y 780 entramos en un pequeño bucle encargado de esperar la entrada de algún carácter por el teclado.

Si pulsamos la tecla de BORRADO, o la de ENTER, el ordenador no obedece y vuelve a la línea 770. Las teclas que sí obedecen al programa son las de «control P», en donde llama a la subrutina de la línea 1060 que da la orden de imprimir, forzando a que dicho texto sea escrito en la impresora; y la tecla ESCAPE, con la que saltamos a la sentencia de la línea 1480 que es «final de programa», donde la máquina recupera sus condiciones normales restableciendo los colores originales, pasa a SCREEN 0 con un ancho de 40 caracteres por línea, deja el color de fondo y borde azul con tinta blanca, borra la pantalla y visualiza las teclas de función, finalizando el programa.

Si lo que entra por el teclado son las teclas de *movimiento de cursor*, ejecuta la función correspondiente.

Si pulsamos la tecla de movimiento de cursor a la derecha, se llamará a la subrutina de la línea 1130, que reescribe el carácter existente sobre fondo blanco haciendo desaparecer el cursor. Incrementa el número de columna si éste es inferior a 25, ya que es el número máximo de caracteres que caben en una línea. Vuelve a calcular nuevamente la posición en la pantalla gráfica, y altera la información reescribiendo sobre fondo gris el carácter ya existente.

En la línea 1200 sucedería el mismo proceso, pero con desplazamiento del cursor a la izquierda hasta llegar a la columna 1 donde quedaría limitado.

La subrutina de la línea 1270 nos permite desplazar el cursor hacia arriba hasta llegar a la fila 1.

Para terminar con las funciones de movimiento del cursor, vamos a la línea 1340 donde lo desplazamos hacia abajo hasta llegar a la fila 6.

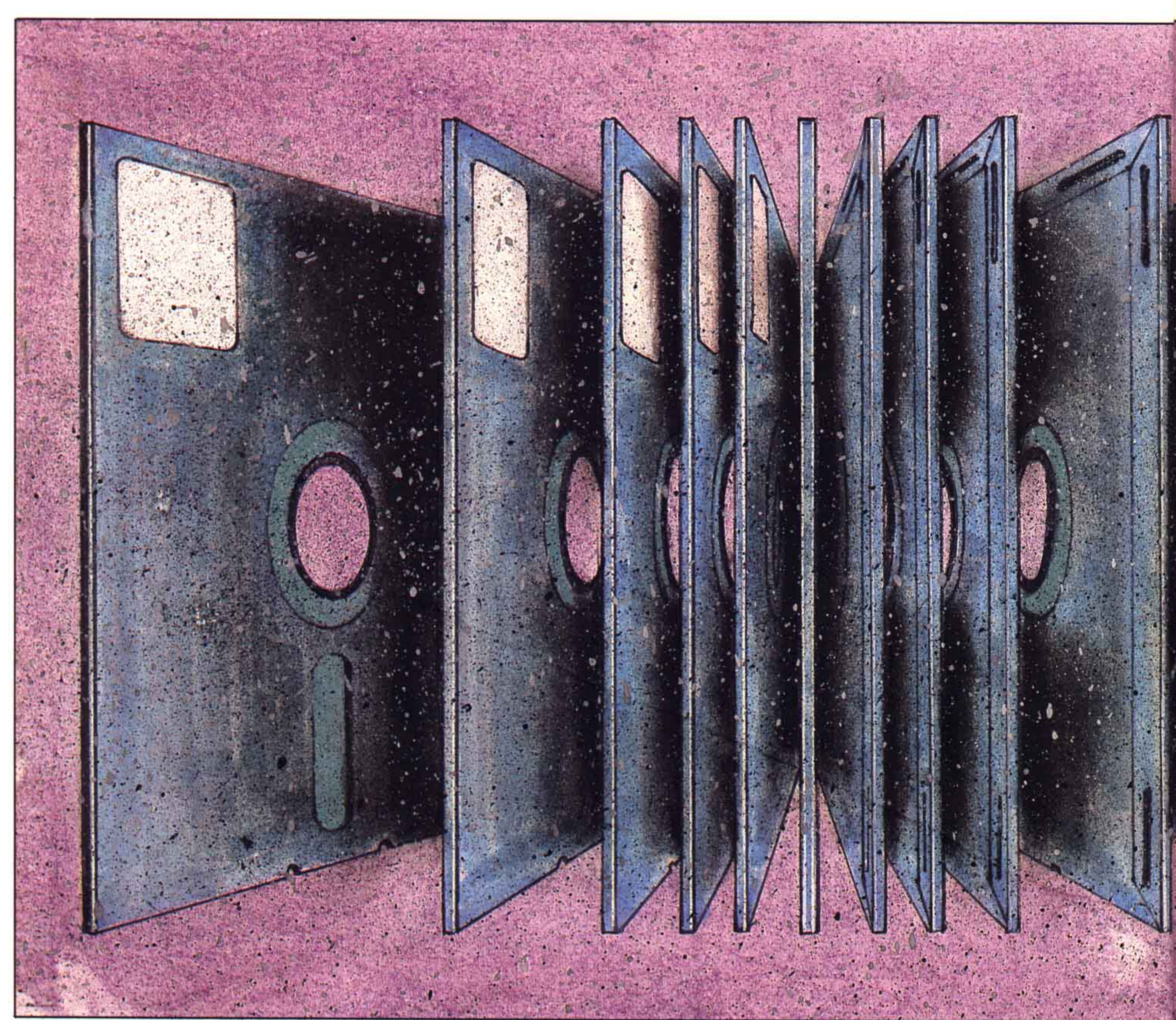
Si el carácter entrado por la consola no es ninguna de las teclas de control ni cualquier otra ya citada, se tratará como un carácter imprimible, pasando a los comandos de la línea 1010 donde entra la información.

Entonces sustituye el carácter anterior por el introducido en la línea 1030, y se dirige a la subrutina que aparece en la línea 1130, que hace avanzar el cursor.

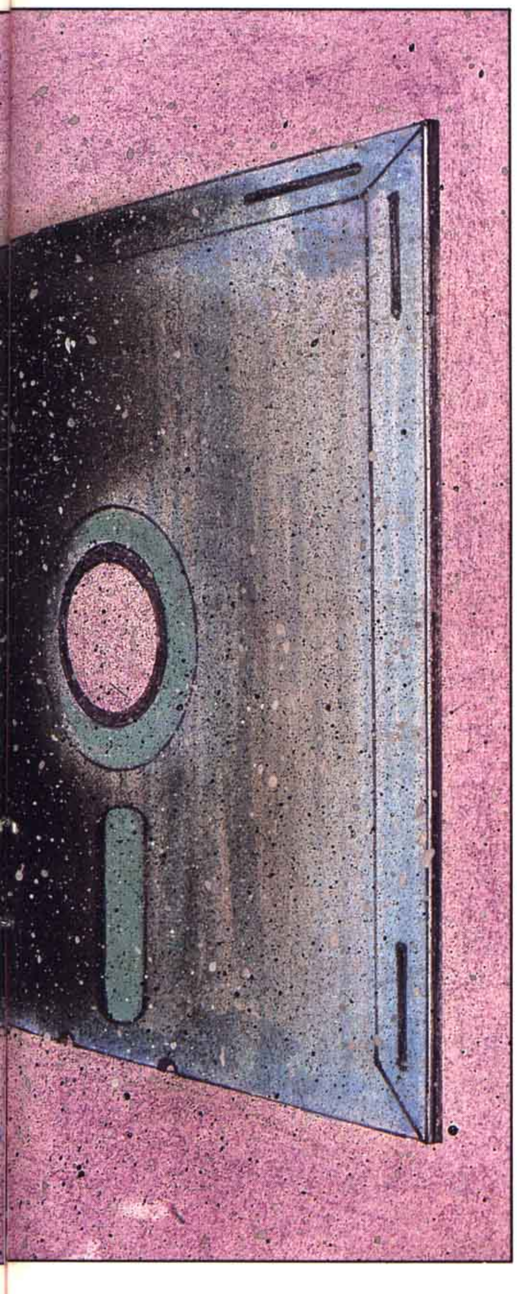
```
100 ' ***********************
110 ' *                     *
120 ' * Programa para escribir *
130 ' * las etiquetas de los   *
140 ' * microfloppys.          *
150 ' *                     *
160 ' * INPUT-MSX J. Garcia.  *
170 ' *                     *
180 ' ***********************
190 '
200 WIDTH 80
210 SCREEN 6
220 DIM T$(6)
230 OPEN "grp:" FOR OUTPUT
```



```
     AS #1
240  'Alteracion de los colores
250  COLOR=(Ø,Ø,Ø,Ø):'Ø=Negro
260  COLOR=(1,Ø,Ø,7):'1=Azul
270  COLOR=(2,5,5,5): '2=Gris
280  COLOR=(3,7,7,7):'3=
     Blanco
290  COLOR 1,2,2 :'azul, gris,
     gris
3ØØ  CLS
31Ø  'Definicion de parametros
32Ø  XO=1Ø1        :'x inicial
33Ø  YO=3Ø         :'y inicial
34Ø  LL=25*8+4 :'long linea
35Ø  OL=Ø          :'offset lineas
36Ø  EL=2*1Ø       :'entre lineas
37Ø  AN=25*8+4 :'ancho total
38Ø  AL=EL*8       :'alto total
39Ø  TX=XO+3       :'x ini text
4ØØ  TY=YO-8       :'y ini text
41Ø  'Dibujo de la etiqueta
42Ø  LINE (XO,YO)-(XO+AN,
     YO+AL),3,BF
43Ø  LINE (XO,YO)-(XO+AN,
     YO+AL),1,B
44Ø  'Rayado interior
45Ø  FOR I=1 TO 6
46Ø  OL=OL+EL
47Ø  LINE (XO,YO+OL)-(XO+LL,
     YO+OL)
48Ø  NEXT I
49Ø  PAINT (XO+1,YO+AL-1)
5ØØ  'Rotulo INDEX
51Ø  PRESET (XO+3,YO+2)
52Ø  COLOR ,3:'fondo blanco
53Ø  PRINT #1, "INDEX"
54Ø  'Otros rotulos
55Ø  COLOR 3,1:'tinta blanca en
     fondo azul
56Ø  PRESET (XO+1,YO+AL-2Ø)
57Ø  PRINT #1,CHR$(2Ø8);CHR$
     (23);" WRITE PROTECT"
```



```
580 PRESET (XO+1,YO+AL
    −10)
590 PRINT #1,CHR$(208);CHR$
    (23);" WRITE ENABLE "
600 COLOR 1,3:'tinta azul, fondo
    blanco
610 'Inicializacion de texto
620 FOR I=1 TO 6
630 T$(I)="Linea de texto
    disponible"
640 NEXT I
650 C=1
660 FOR F=1 TO 6
670 GOSUB 1420
```



```
680 PRINT #1,T$(F);
690 NEXT F
700 '
710 'gestion de entrada de datos
720 '--------------------------
730 F=1:C=1:GOSUB 1420
740 COLOR 1,2:'tinta azul en
    papel gris
750 PRINT #1,MID$(T$(F),C,1);
760 GOSUB 1420
770 A$=INKEY$
780 IF A$="" GOTO 770
790 IF A$=CHR$(8) GOTO
    770:'rub
800 IF A$=CHR$(13) GOTO
    770:'ret
810 IF A$=CHR$(16) THEN
    GOSUB 1060:GOTO 770 :'
    impr
820 IF A$=CHR$(27) GOTO
    1480:'esc
830 IF A$=CHR$(28) THEN
    GOSUB 1130:GOTO 770:
    'der
840 IF A$=CHR$(29) THEN
    GOSUB 1200:GOTO 770
    :'izq
850 IF A$=CHR$(30) THEN
    GOSUB 1270:GOTO 770
    :'sub
860 IF A$=CHR$(31) THEN
    GOSUB 1340:GOTO 770
    :'baj
870 GOSUB 1010 :'entrar
    caracter
880 GOTO 770 :'repite bucle
890 'restaura la informacion
900 '----------------------
910 COLOR 1,3
920 PRINT #1,MID$(T$(F),C,1);
930 GOSUB 1420
940 RETURN
950 'altera el texto
960 '--------------
970 COLOR 1,2
980 PRINT #1,MID$(T$(F),C,1);
990 GOSUB 1420
1000 RETURN
1010 'entrar nuevo texto
1020 '-----------------
1030 MID$(T$(F),C,1)=A$
1040 GOSUB 1130
1050 RETURN
1060 'imprimir todo el texto
1070 '---------------------
1080 FOR I=1 TO 6
1090 LPRINT T$(I)
1100 LPRINT
1110 NEXT I
1120 RETURN
1130 'avanzar el cursor
1140 '----------------
1150 GOSUB 890
1160 IF C<25 THEN C=C+1
1170 GOSUB 1420
1180 GOSUB 950
1190 RETURN
1200 'retroceder el cursor
1210 '-------------------
1220 GOSUB 890
1230 IF C>1 THEN C=C−1
1240 GOSUB 1420
1250 GOSUB 950
1260 RETURN
1270 'subir el cursor
1280 '--------------
1290 GOSUB 890
1300 IF F>1 THEN F=F−1
1310 GOSUB 1420
1320 GOSUB 950
1330 RETURN
1340 'bajar el cursor
1350 '--------------
1360 GOSUB 890
1370 IF F<6 THEN F=F+1
1380 GOSUB 1420
1390 GOSUB 950
1400 RETURN
1410 '
1420 'calculo de x,y grafica
1430 '--------------------
1440 X=TX+(8*(C−1))
1450 Y=TY+(EL*F)
1460 PRESET (X,Y)
1470 RETURN
1480 'fin de programa
1490 '--------------
1500 COLOR=NEW
1510 SCREEN 0
1520 WIDTH 40
1530 COLOR 15,4,4
1540 CLS
1550 KEY ON
1560 END
```

TU PRIMER FRONTON

- UNA PARTIDA DE FRONTON
- UN DIFICIL ADVERSARIO: TU ORDENADOR
- ANALISIS DEL PROGRAMA

Muchos de los lectores de INPUT debéis añorar aquellos primeros videojuegos frente a los que se pasaban largos ratos jugando a tenis, ping-pong... Este sencillo programa que ahora os presentamos os permitirá jugar una amena partida de frontón contra vuestro ordenador.

En primer lugar os vamos a explicar el funcionamiento del programa. Éste se inicia en la línea 10, con la que se abre un fichero, gracias al cual podremos escribir caracteres en una pantalla de gráficos. La línea siguiente, la 20, define algunas variables como las de puntuación o número de bolas, etc... En la línea 30 se inicializa la pantalla y es desde ella que se nos pregunta el nivel de juego, guardando la respuesta en la variable N.

En la línea 80 se pregunta por la velocidad de nuestra raqueta y la guarda en V. La línea 100 determina la velocidad dependiendo del nivel. La línea 110 prepara el color de la pantalla. La 120 es la que autoriza las interrupciones por choque de sprites.

A continuación, la línea 130 selecciona el modo de gráficos de alta resolución, mediante el uso de sprites ampliados. La línea 140 utiliza una rutina del bios encargada de desactivar la pantalla. Desde la línea 150 hasta la 180 se imprimen, aleatoriamente, los espectadores. Mientras que desde la línea 190 a la 230 se dibuja la pista, así como los indicadores de bolas y de puntuación.

La línea 240 utiliza la rutina del bios, que complementa a la utilizada en la línea 140, activando la pantalla. Desde la línea 250 a la 330 se diseñan los sprites de la bola y de la paleta. Las líneas 340 y 350 eligen la dirección aleatoria de salida de la bola. La línea 360 define la posición inicial de la bola. La 370 define la posición de la paleta, utilizando, según el caso, las rutinas entre las líneas 430 a 620.

La línea 630 visualiza la bola en la pantalla y desde la 640 hasta la 690 el ordenador recibe datos desde el joystick y del teclado, a la vez que se encarga de que la paleta no se salga de la pista. La línea 700 visualiza la paleta.

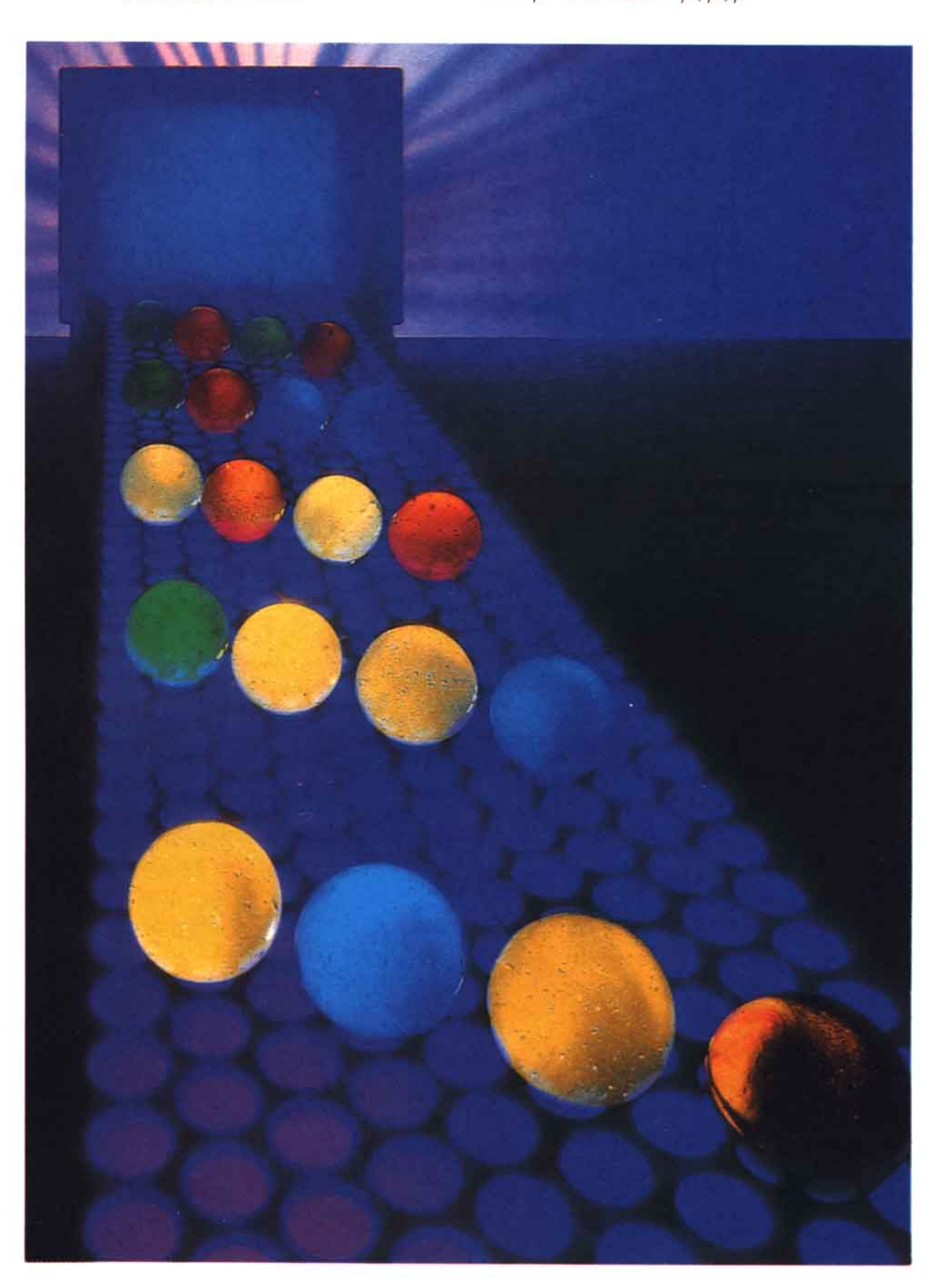
Entre las líneas 710 a 740 se comprueba si la bola ha chocado con alguna de las paredes o ha sido interceptada por nuestra paleta, en cuyo caso el ordenador manda el control a la rutina correspondiente situada entre las líneas 760 y 1010. Esta rutina se encarga de hacer rebotar la bola. Si ha sido un golpe nuestro con la paleta el que ha hecho rebotar la bola la rutina anterior es llamada desde la línea 650. Desde la línea 1110 hasta la 1200 se programa la despedida así como la opción de volver a jugar o retirarte.

Al ser un programa en Basic, de lenta velocidad de ejecución, una vez hayas introducido la orden de ejecución y los datos sobre el nivel de dificultad y la velocidad de la paleta, el ordenador dejará durante unos segundos la pantalla desconectada antes de proseguir el juego.

Adelante y a ganar.

```
1Ø OPEN"GRP:" AS #1
2Ø G=1:H=1:NB=3:PT=Ø
3Ø CLS:KEY OFF:COLOR 3,1,1:INPUT "NIVEL(1-7)";N
4Ø IF N<1 OR N>7 THEN GOTO 3Ø
5Ø IF N<=3 THEN L=2
6Ø N=>4 AND N<6 THEN L=4
7Ø IF N=7 OR N=6 THEN L=6
8Ø INPUT "VELOCIDAD DE LA PALETA:(1-4)";V
9Ø IF V<1 OR V>4 THEN GOTO 8Ø
1ØØ Z1=N/2:Z2=N
11Ø COLOR 3,1,1
12Ø SPRITE ON
13Ø SCREEN 2,3
14Ø DEF USR=&H41:J=USR(Ø)
15Ø FOR S=1 TO 16Ø
16Ø PSET(G,H),1:HH=INT(RND(1)*2)+1:IF HH=1 THEN PRINT #1,"[GRAF+{]" ELSE PRINT #1,"[GRAF+SHIFI+{]"
17Ø G=G+8:IF G=>25Ø THEN H=H+7:G=1
18Ø NEXT S
19Ø LINE (4Ø,4Ø)-(24Ø,14Ø),3,BF
2ØØ PSET (1Ø,175),1
21Ø PRINT #1,"PUNTUACION: Ø"
22Ø PSET (15Ø,175),1
23Ø PRINT #1,"BOLAS:3"
24Ø DEF USR=&H44:J=USR(Ø)
25Ø A$=CHR$(&B1)
26Ø SPRITE$(Ø)=A$
27Ø B1$=CHR$(&B11)
28Ø B2$=CHR$(&BØØ)
29Ø B3$=CHR$(&B11)
3ØØ B4$=CHR$(&BØØ)
31Ø B5$=CHR$(&B11)
32Ø B$=B1$+B2$+B3$+B4$+B5$
33Ø SPRITE$(1)=B$
34Ø HH=INT(RND(1)*2)+1
35Ø IF HH=1 THEN D$="D3" ELSE D$="D4"
36Ø X=221:Y=7Ø
37Ø A=7Ø:B=8Ø
38Ø REM DIRECCIONES
39Ø IF D$="D1"THEN GOTO 43Ø
4ØØ IF D$="D2"THEN GOTO 48Ø
41Ø IF D$="D3"THEN GOTO 53Ø
42Ø IF D$="D4"THEN GOTO 58Ø
43Ø REM D1
```

```
440 X=X+Z2
450 Y=Y-Z1
460 GOTO 630
470 REM --------
480 REM D2
490 X=X+Z2
500 Y=Y+Z1
510 GOTO 630
520 REM --------
530 REM D3
540 X=X-Z2
550 Y=Y-Z1
560 GOTO 630
570 REM --------
580 REM D4
590 X=X-Z2
600 Y=Y+Z1
610 GOTO 630
620 REM --------
630 PUT SPRITE 0,(X,Y),1,0
640 D=STICK(0)
650 ON SPRITE GOSUB 1020
660 IF D=1 THEN B=B-V
670 IF D=5 THEN B=B+V
680 IF B=<40 THEN B=40
690 IF B=>130 THEN B=130
700 PUT SPRITE 1,(70,B),15,1
710 IF X=<60 THEN GOTO 760
720 IF X=>225 THEN GOTO
    900
730 IF Y=<40 THEN GOTO 840
740 IF Y=>140 THEN GOTO
    960
750 GOTO 380
760 PLAY "140v15o6fo3c"
770 NB=NB-1
780 LINE (195,170)-(255,
    185),1,BF
790 PSET (195,175),1
800 PRINT #1,NB
810 IF NB<>0 THEN GOTO 830
820 GOTO 1110
830 GOTO 340
840 ' RUTINA ANTICHOQUE B
850 PLAY "L63V15O5GC"
860 IF D$="D3"THEN D$="D4"
870 IF D$="D1"THEN D$="D2"
880 Y=Y+2
890 GOTO 630
900 ' RUTINA ANTICHOQUE C
910 PLAY "L63V15O5GC"
920 IF D$="D1"THEN D$="D3"
930 IF D$="D2"THEN D$="D4"
940 X=X-2
950 GOTO 630
960 ' RUTINA ANTICHOQUE D
970 PLAY "L63V15O5GC"
980 IF D$="D4"THEN
    D$="D3"
990 IF D$="D2"THEN
    D$="D1"
1000 Y=Y-2
1010 GOTO 630
1020 PLAY
     "V15L63O4CO5
     CO6CO7C"
1040 IF D$="D4" THEN
     D$="D2"
1050 PT=PT+5*N
1060 LINE(95,170)-(150,185),
     1,BF
1070 PSET (95,175),1:PRINT
     #1,PT
1080 SPRITE ON
1090 PUT SPRITE 0,
     (X+L,Y),0,0
1100 GOTO 380
1110 SCREEN 0,0,0
```



PON COLOR EN TUS PROGRAMAS

El programa que os presentamos a continuación os permitirá dibujar en la pantalla de vuestro ordenador hasta casi 4.000 puntos, utilizando los dieciséis colores. Pero la mayor ventaja que ofrece el programa es la de poder grabar el dibujo en cinta.

El programa está estructurado en tres partes claramente definidas. La primera, por orden de programación, es la presentación que, en realidad, es la menos importante de las tres. Se extiende entre las líneas 10 a 330.

A partir de aquí comienza el programa en sí. Desde las líneas 340 a la 410 se inicializa la pantalla y se definen algunas variables. A partir de la 420, hasta la línea 500, se extiende el primer menú, que es quien nos muestra la posibilidad de manejar la paleta mediante el cursor o alternativamente con el joystick, para lo que la variable J recoge el valor entrado en la línea 500. Este valor será el que utilice más tarde la sentencia STICK.

La línea 510 nos abre un archivo con el que podemos escribir caracteres en la pantalla de gráficos. El segundo menú se desarrolla entre las líneas 520 y 650 y es el encargado de mostrarnos las posibilidades de esta paleta gráfica: borrar la pantalla, subir y bajar la pluma, cambiar colores, etc...

Desde las líneas 670 hasta la 960 se dibujan la paleta y el panel de colores. A partir de la instrucción 970 se crea y define un sprite de 8*8, que será quien más tarde representará la pluma. La línea 1060 se encargará de situar la pluma en el centro de la pantalla, cuyo movimiento se define entre las líneas 1070 a 1180. Las cuatro siguientes limitan el movimiento de la pluma para que no salga de la pantalla.

Desde este punto y hasta la línea 1280, el ordenador queda a la espera de recibir instrucciones desde el teclado. La línea 1290 se encarga de que los datos de cada punto dibujado, que se graban entre las posiciones de memoria 49152 y 61000, no rebasen esta última a fin de que no se solapen con el área del sistema.

Las líneas 1300 a 1320 se encargan de la visualización de la pluma y de los puntos que conforman el dibujo. Las líneas 1330 y 1340 comprueban que ese punto no ha sido dibujado antes, para no archivarlo, y de guardar los datos cuando se trata de un nuevo punto. La línea 1360 devuelve el control a 1090, para que siga leyendo los datos del teclado.

En 1370 se inicia la rutina de cambio de color, situando la pluma en el panel de colores. Entre 1490 y 1510 espera a que pulsemos N o P, para elegir un nuevo color o para volver al panel. 1520 visualiza la pluma y 1530 repite todo el proceso de cambio de color.

La rutina de retorno al dibujo, definida entre las líneas 1560 y 1610, se encarga de repetir lo dibujado hasta el momento y representarlo en la pantalla al abandonar el menú de aplicaciones. Para ello extrae los valores pokeados anteriormente a partir de la posición de memoria 49152 y los utiliza en una sentencia PSET. La línea 1610 retorna el control a 680.

A partir de la línea 1620 se extiende el menú de grabación, con el que podemos elegir entre grabar, extraer del cassette o volver al dibujo.

La rutina de grabación empieza en 1680, donde graba los datos de cada punto en un fichero llamado GRAF. Al terminar, el ordenador muestra dos opciones: anular el dibujo o volver a éste. En 1830 se inicia la rutina de extracción en la que se abre el mismo fichero GRAF y se recogen los datos sobre cada punto. Al terminar obtienes de nuevo dos opciones, repetir la operación o presentar en pantalla el dibujo recogido.

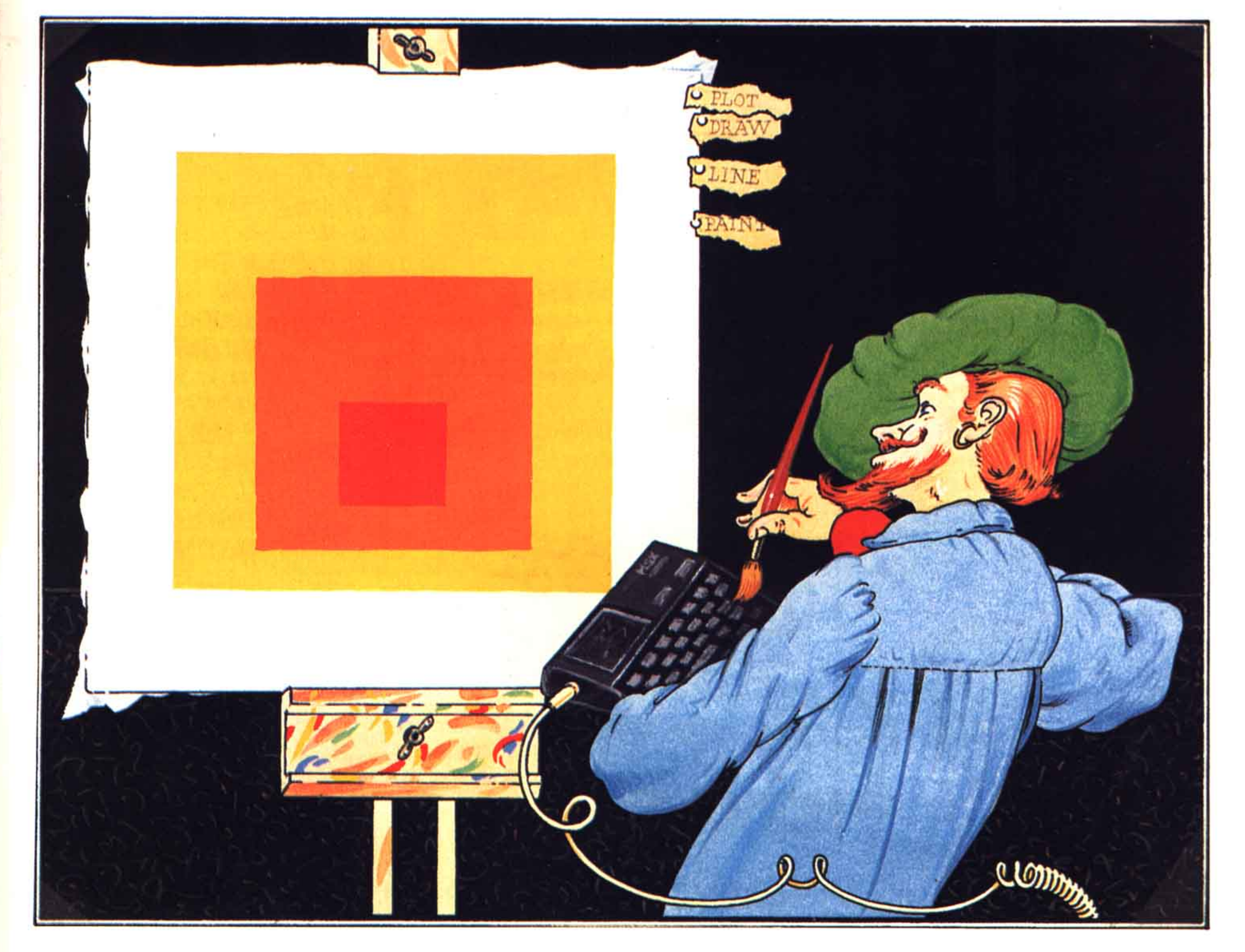
Una vez hayan aparecido en pantalla la paleta gráfica y el panel de colores si manejáis el programa desde el teclado, debéis apretar las siguientes teclas: con W la pluma tocará el papel y podréis dibujar, con Q la pluma se elevará, con B borraréis la pantalla, si eliges G podrás cambiar de color (para ello debes situar la pluma sobre el color elegido y apretar N), P para volver al papel y, finalmente, G para grabar el dibujo.

La paleta gráfica se debe usar con mayúsculas.

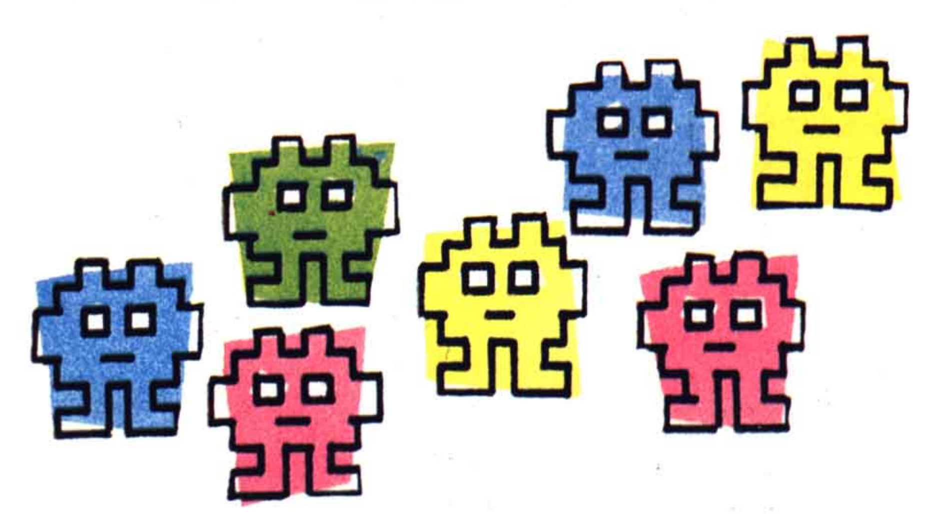
Después de trabajar con este programa ya podréis desarrollar libremente vuestras inclinaciones pictóricas, bien dibujando o construyendo paisajes y portadas para vuestros juegos.

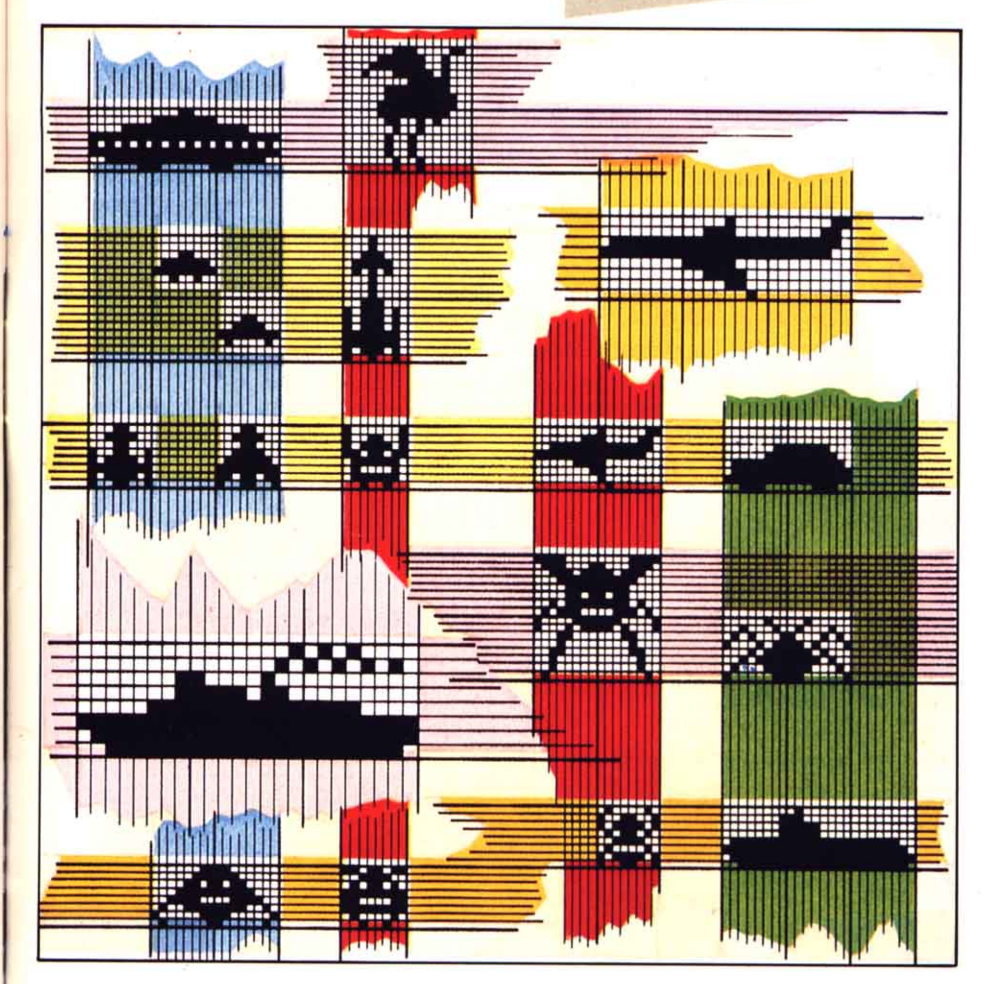
```
10 F=60
20 COLOR 3, 1, 1
30 SCREEN 2, 2, 0
40 OPEN "GRP:" AS #1
50 REM PRESENTACION
60 FOR T=70 TO 100
70 PSET (F, T), 1
80 PRINT #1, "PALETA
   GRAFICA"
90 F=F+1
100 COLOR ,,INT(RND(1) *
    15)+1
110 NEXT T
120 PSET (F+5, T+6), 3
130 PRINT #1, "PALETA
    GRAFICA"
140 PSET (F+6, T+7), 1
150 PRINT #1, "PALETA
    GRAFICA"
160 PSET (30, 140), 1
170 PRINT #1, "INPUT"
180 PSET (31, 141), 1
190 PRINT #1, "INPUT"
200 COLOR 8, 1, 1
210 F=140
220 FOR T=100 TO 110
230 PSET (T, F), 1
```

```
240 PRINT #1, "M.S.X."
250 F=F+1
260 NEXT T
270 PSET (T+4, F+4), 1
280 PRINT #1, "M.S.X."
290 PSET (T+5, F+5), 1
300 PRINT #1, "M.S.X."
310 CLOSE #1
320 IF INKEY$<> " " THEN
    GOTO 340
330 GOTO 320
340 MAXFILES=2
350 H=49152!
360 REM PALETA GRAFICA
370 REM
380 REM MENU DE CONTROL
390 CLS:KEY OFF:COLOR 3, 1,
    1:SCREEN 0,0,0:WIDTH
    35:CA=15:PL=1
400 POKE H, 0:POKE H+1,
    0:POKE H+2, 0
410 H=H+3
420 REM
430 PRINT "MENU DE
    CONTROLES"
440 PRINT:PRINT:PRINT
450 PRINT "(0) CONTROL POR
    CURSOR"
460 PRINT "(1) CONTROL POR
    JOYSTICK(1);;
470 PRINT "(2) CONTROL POR
    JOYSTICK(2)"
480 "--------------------------------"
```



```
49Ø PRINT:PRINT:PRINT:PRINT
5ØØ INPUT "Escoja la opcion
    deseada."; J
51Ø OPEN "GRP:" AS #1
52Ø CLS
53Ø PRINT "DIRECTORIO DE
    APLICACIONES"
54Ø PRINT:PRINT:PRINT
55Ø PRINT "(B) Borrado de
    pantalla"
56Ø PRINT "(Q) Sube la pluma"
57Ø PRINT "(W) Baja la pluma"
58Ø PRINT "(C) Cambia colores"
59Ø PRINT "      —N— Elige
    nuevo color"
6ØØ PRINT "      —P— Retorna
    al dibujo"
61Ø PRINT "(R) Retorna a este
    menu"
62Ø PRINT "(G) I/O Dibujo a
    cinta"
63Ø PRINT:PRINT:PRINT
64Ø PRINT "------------------------"
65Ø IF INKEY$ <> " " THEN
    GOTO 156Ø
66Ø GOTO 65Ø
67Ø SCREEN 2, 2, Ø
68Ø REM RETORNO
69Ø COLOR 8
7ØØ LINE (1Ø, 15)–(2ØØ, 185),
    1Ø, B
71Ø PSET (1Ø, 7), 1
72Ø PRINT #1, «Paleta Grafica"
73Ø PSET (11, 7), 1
74Ø PRINT #1, "Paleta Grafica"
75Ø COLOR 3
76Ø LINE (21Ø, 15) – (22Ø, 25),
    2, BF
77Ø LINE (23Ø, 15) – (24Ø, 25),
    3, BF
78Ø LINE (21Ø, 35) – (22Ø, 45),
    4, BF
79Ø LINE (23Ø, 35) – (24Ø, 45),
    5, BF
8ØØ LINE (21Ø, 55) – (22Ø, 65),
    6, BF
81Ø LINE (23Ø, 55) – (24Ø, 65),
    7, BF
82Ø LINE (21Ø, 75) – (22Ø, 85),
    8, BF
83Ø LINE (23Ø, 75) – (24Ø, 85),
    9, BF
84Ø LINE (21Ø, 95) – (22Ø,
    1Ø5), 1Ø, BF
85Ø LINE (23Ø, 95) – (24Ø,
    1Ø5), 11, BF
86Ø LINE (21Ø, 115) – (22Ø,
    125), 12, BF
87Ø LINE (23Ø, 115) – (24Ø,
    125), 13, BF
88Ø LINE (21Ø, 135) – (22Ø,
    145), 14, BF
89Ø LINE (23Ø, 135) – (24Ø,
    145), 15, BF
9ØØ LINE (21Ø, 155) – (22Ø,
    165), 15, B
91Ø LINE (2Ø5, 1Ø) – (25Ø,
    17Ø), 1Ø, B
92Ø PSET (2ØØ  , 2), 1
93Ø PRINT #1, "Colores"
94Ø PSET (2Ø1, 2), 1
95Ø PRINT #1, "Colores"
96Ø LINE (1Ø, 186) – (2ØØ,
    189), CA, BF
97Ø A1$=CHR$(&OØBØØØ1ØØØØ)
98Ø A2$=CHR$(&OØBØØØ1ØØØØ)
99Ø A3$=CHR$(&OØBØØØØØØØØ)
1ØØØ A4$=CHR$(&OØB11ØØØ11Ø)
1Ø1Ø A5$=CHR$(&OØBØØØØØØØØ)
1Ø2Ø A6$=CHR$(&OØBØØØ1ØØØØ)
1Ø3Ø A7$=CHR$(&OØBØØØ1ØØØØ)
1Ø4Ø A$=A1$+A2$+A3$+A4$
     +A5$+A6$+A7$
1Ø5Ø SPRITE$(Ø)=A$
1Ø6Ø X=1ØØ :Y=89
1Ø7Ø REM PLUMA
1Ø8Ø D=STICK(J)
1Ø9Ø K$=INKEY$
11ØØ IF D=1 THEN Y=Y–1
111Ø IF D=2 THEN
     X=X+1:Y=Y–1
112Ø IF D=3 THEN X=X+1
113Ø IF D=4 THEN
     X=X+1:Y=Y+1
114Ø IF D=5 THEN Y=Y+1
115Ø IF D=6 THEN X=X–
     1:Y=Y+1
116Ø IF D=7 THEN X=X–1
117Ø IF D=8 THEN X=X–
     1:Y=Y–1
118Ø IF D=Ø THEN X=X:Y=Y
119Ø IF X=>196 THEN X=196
12ØØ IF X=<8 THEN X=8
121Ø IF Y=>18Ø THEN Y=18Ø
122Ø IF Y=<12 THEN Y=12
123Ø IF K$="C" THEN GOTO
     137Ø
124Ø IF K$="B" THEN LINE
     (11, 16)–(199, 184), 1,
     BF:H=49155!
125Ø IF K$="Q" THEN PL=1
126Ø IF K$="W" THEN PL=Ø
127Ø IF K$="R" THEN GOTO
     154Ø
128Ø IF K$="G" THEN GOTO
     162Ø
129Ø IF H=>61ØØØ! THEN
     CIRCLE (18Ø, 3), 5, 8,,,
     1.4:PAINT (18Ø, 3),
     8:GOTO 136Ø
13ØØ PUT SPRITE Ø, (X, Y), 15,
     Ø
131Ø IF PL=1 THEN GOTO 136Ø
132Ø PSET (X+3, Y+4), CA
133Ø IF PEEK(H–3)=X+3 AND
     PEEK(H–2)=Y+4 AND
```





```
     PEEK(H–1)=CA THEN
     GOTO 136Ø
134Ø POKE H, X+3:POKE H+1,
     Y+4:POKE H+2, CA
135Ø H=H+3
136Ø GOTO 1Ø8Ø
137Ø REM RUTINA DE CAMBIO
     DE COLOR
138Ø X=225:Y=2Ø
139Ø D=STICK(J)
14ØØ K$=INKEY$
141Ø IF D=1 THEN Y=Y–1
142Ø IF D=3 THEN X=X+1
143Ø IF D=5 THEN Y=Y+1
144Ø IF D=7 THEN X=X–1
145Ø IF X=>246 THEN X=246
146Ø IF X=<2Ø3 THEN X=2Ø3
147Ø IF Y=<7 THEN Y=7
148Ø IF Y=>165 THEN Y=165
149Ø IF K$="N" THEN
     CA=POINT (X+3, Y+4)
15ØØ IF K$="N" THEN LINE
     (1Ø, 186)–(2ØØ , 189),
     CA, BF
151Ø IF K$="P" THEN GOTO
     1Ø6Ø
152Ø PUT SPRITE Ø, (X, Y), 15,
     Ø
153Ø GOTO 139Ø
154Ø REM RUTINA DE
     RETORNO AL MENU
155Ø SCREEN Ø:CLS:GOTO 52Ø
156Ø REM RUTINA DE
     REGRESO AL DIBUJO
157Ø SCREEN 2, 2, Ø
158Ø FOR WQ=49155! TO H–1
     STEP 3
159Ø PSET (PEEK(WQ),
     PEEK(WQ+1)),
     PEEK(WQ+2)
16ØØ NEXT WQ
161Ø GOTO 68Ø
162Ø CLS:SCREEN Ø, Ø, Ø:REM
     I/O CASETE
163Ø INPUT "(1)GRABAR
     DIBUJO(2) EXTRAER
     DIBUJO(3) VOLVER AL
     DIBUJO"; L
164Ø IF L=1 THEN GOTO 168Ø
165Ø IF L=2 THEN GOTO 183Ø
166Ø IF L=3 THEN GOTO 156Ø
167Ø GOTO 163Ø
168Ø REM RUTINA DE
     GRABACION
169Ø OPEN "CAS:GRAF" FOR
     OUTPUT AS #2
17ØØ PRINT #2, H
171Ø FOR WQ=49152! TO H–1
172Ø PRINT #2, PEEK(WQ)
173Ø NEXT WQ
174Ø CLOSE #2
175Ø PRINT:PRINT
176Ø PRINT "La grabacion ha
     terminado"
177Ø PRINT "quieres anular el
     dibujo–1–"
178Ø PRINT "o prefieres volver al
     dibujo–2–"
179Ø INPUT YH
18ØØ IF YH=1 THEN
     H=49155!:GOTO 52Ø
181Ø IF YH=2 THEN GOTO
     156Ø
182Ø GOTO 175Ø
183Ø CLS:REM RUTINA DE
     EXTRACCION
184Ø OPEN "CAS:GRAF" FOR
     INPUT AS #2
185Ø INPUT #2, H
186Ø FOR WQ=49152! TO H–1
187Ø INPUT #2, A
188Ø POKE WQ, A
189Ø NEXT WQ
19ØØ CLOSE #2
191Ø PRINT "La extraccion ha
     terminado"
192Ø PRINT "quieres realizar
     otra extraccion–1–"
193Ø PRINT "o prefieres
     representar en pantalla–2–
     "
194Ø INPUT RE
195Ø IF RE=1 THEN GOTO
     183Ø
196Ø IF RE=2 THEN GOTO
     156Ø
197Ø GOTO 191Ø
198Ø END
```

EL BOSQUE MALDITO

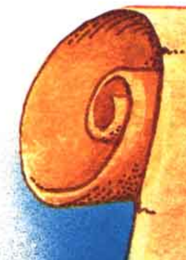
Este mes en la sección de PARTICIPA os presentamos un programa que, a despecho de su sencillez y reducida extensión, configura un juego electrizante que obliga al jugador a mantener sus nervios en tensión de principio a fin. Será muy difícil que consigas salvarte de la presión del fantasma o que no te estrelles contra los macizos muros que cierran el bosque, en un intento frustrado por huir de las fauces del fantasmagórico poblador del bosque. Si pisas tus propias huellas también perecerás. Tus movimientos por el bosque se dirigen muy fácilmente desde las teclas de movimiento del cursor.

```
1Ø ' *********************************
2Ø '** .BOSQUE MALDITO/ POR A.F.B. *
3Ø ' *********************************
4Ø DIMSC(1Ø),N$(1Ø):GOSUB49Ø
5Ø '***** INICIACION ********
6Ø COLOR 1,1Ø,12:SCREEN1,2:KEYOFF:WIDTH 32:Z=8:SC=Ø:LF=2:Q=Ø:W=6144+32*
   18+2Ø:R=1ØØ:T=1ØØ:ONSPRITEGOSUB42Ø:SPRITEON:X=16Ø:Y=143:ONSTOPGOSUB45Ø:
   STOPON:F=2Ø:G=8Ø:ONINTERVAL=ZGOSUB38Ø:L=4:C=4
7Ø '***** DEF. SPRITES ****
8Ø RESTORE8Ø:FORA=14336TO14336+32*4-1:READA$:VPOKEA,VAL
   ("&H"+A$):NEXT:DATA18,18,3C,3C,7E,7E,FF,FF,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,
   Ø,Ø,Ø,Ø,CØ,FØ,FC,FF,FF,FC,FØ,CØ,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,FF,FF,
   7E,7E,3C,3C,18,18
9Ø DATA Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,3,F,3F,FF,FF,3F,F,3,Ø,Ø,Ø,Ø,Ø,Ø,
   Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
1ØØ DATA7,7F,6F,5F,7D,F5,E5,D5,DØ,CØ,CØ,68,6A,2A,1A,7,EØ,FE,F6,FA,BE,AF,A7,AB,B,3,3,
    16,56,54,58,EØ:RESTORE1ØØ:FORA=ØTO31:READA$:B$=B$+CHR$(VAL("&H"+A$)):
    NEXT:SPRITE$(4)=B$
11Ø B$="":DATA1,1,1,21,12,6,15,FA,38,6,E,11,21,1,1,1,Ø,Ø,8Ø,8Ø,84,C8,3Ø,AE,9F,98,7Ø,
    E,9,88,Ø,Ø,Ø:RESTORE12Ø:FORA=ØTO31:READA$:B$=B$+CHR$(VAL("&H"+A$)):
    NEXT:SPRITE$(6)=B$
12Ø B$="":DATA1,1,1,21,12,6,15,FA,38,6,E,11,21,1,1,1,Ø,Ø,8Ø,8Ø,84,C8,3Ø,AE,9F,98,7Ø,
    E,9,88,Ø,Ø,Ø:RESTORE12Ø:FORA=ØTO31:READA$:B$=B$+CHR$(VAL("&H"+A$)):
    NEXT:SPRITE$(6)=B$
13Ø X=16Ø:Y=143:W=6144+32*18+2Ø:R=1ØØ:T=1ØØ:F=2Ø:G=8Ø:SPRITEON
14Ø '***** IMPRIMIR PANTALLA ***
15Ø LOCATE Ø,Ø:PRINT"PUNTOS:":LOCATE12,Ø:PRINT"VIDAS:";SPC(3);"TIEMPO";:LOCATEØ,
    1:PRINTSTRING$(32,"W");:FORA=2TO21:LOCATEØ,A:PRINT"W";SPC(3Ø);"W";:NEXT:
    LOCATEØ,22:PRINTSTRING$(32,"W");SPC(6);"POR A.F.B. 1986";:BEEP
16Ø '** INICIAR SONIDO *******
17Ø K=TIME:FORA=ØTO3Ø:LOCATERND(-TIME)*3Ø+1,RND(K)*18+2:
    PRINT"$":NEXT:TIME=Ø:INTERVALON:SOUND8,16:SOUND9,14:SOUND1Ø,14:SOUND13,Ø:
    SOUND11,68:SOUND12,8:SOUND7,&B1ØØØ1:INTERVALON:GOSUB25Ø
18Ø COLOR1,1Ø,12:VPOKEBASE(6)+4,&HB2:VPOKEBASE(6)+26,&H1E
19Ø IFZ>4THENK=TIME:FORA=ØTO2Ø:LOCATERND(-TIME)*3Ø+1,RND(K)*18+2:
    PRINT"F":NEXT:TIME=Ø:LOCATEØ,1:PRINT"W";
2ØØ PUTSPRITE1,(X,Y),15,Q:SOUND4,X:SOUND2,Y:PUTSPRITE2,(F,G),C,4:
    GOSUB27Ø:INTERVALON:SOUND6,(F+G)/2:IFTIME/5Ø>3ØTHENELSE2ØØ
21Ø INTERVALOFF:BEEP:PLAY"SØM1ØØØØL8O5FRERL2D","SØM1ØØØØL8O5CRCRO4L2B",
```

```
"SØM1ØØØØL8O4ARGRL2G":LOCATE6,1Ø:PRINT"---FASESUPERADA---";
:FORA=ØTO1ØØØ:NEXT:Z=Z-1:IFZ<4THENZ=4:C=RND(1)*1Ø+4:
ONINTERVAL=ZGOSUB38Ø:GOTO13Ø
22Ø C=RND(1)*1Ø+4:ONINTERVAL=ZGOSUB38Ø:GOTO13Ø
23Ø GOTO 23Ø
24Ø '****IMPRIMIR TIEMPO *****
25Ø INTERVALOFF:LOCATE 6,Ø:PRINTSC:LOCATE18,Ø:PRINTLF:LOCATE26,Ø:PRINTINT(TIME/
5Ø);:INTERVALON:RETURN
26Ø '**** LEER STICK Y PUNTOS ***
27Ø INTERVALON:D=STICK(Ø):IFD=1THENGOSUB32ØELSEIFD=3THENGOSUB33ØELSEIFD=
5THENGOSUB34ØELSEIFD=YTHENGOSUB35Ø
28Ø IFVPEEK(W)=36THENSC=SC+1Ø:FORS=ØTO8:INTERVALOFF:NEXT:GOSUB25Ø
29Ø IF VPEEK(W)=&HD7ORVPEEK(W)=&HDBORVPEEK(W)=6THENGOSUB42Ø
3ØØ RETURN
31Ø '**** MOVER FLECHA *****
32Ø Y=Y-8:Q=Ø:VPOKEW,&HDB:W=W-32:RETURN
33Ø X=X+8:Q=1:VPOKEW,&HDB:W=W+1:RETURN
34Ø Y=Y+8:Q=2:VPOKEW,&HDB:W=W+32:RETURN
35Ø X=X-8:Q=3:VPOKEW,&HDB:W=W-1:RETURN
36Ø SC=SC+1Ø:GOSUB25Ø:RETURN
37Ø '*** MOVER FANTASMA ****
```



```
380 IFF<XTHENF=F+3ELSEF=F-3
390 IFG<YTHENG=G+3ELSEG=G-3
400 SOUND13,0:LOCATE26,0:PRINTINT(TIME/50):RETURN
410 '***** MUERTE *****
420 SPRITEOFF:INTERVALOFF:FORA=0TO15:PUTSPRITE1,,A,Q:BEEP:COLOR,,A:
    NEXT:GOSUB440:LF=LF-1:IFLF=0THEN470ELSECOLOR,,1:FORA=0TO100:PUTSPRITE0,
    (VPEEK(6917)-4,VPEEK(6916)-4),RND(1)*15,6:SOUND6,A:NEXT
430 PUTSPRITE0,(0,-16):PUTSPRITE1,(30,-16):RETURN130
440 SOUND7,&B00111:SOUND8,16:SOUND11,88:SOUND12,200:SOUND13,0:RETURN
450 SCREEN0:LIST10-30
460 '**** FIN ##########
470 COLOR3,1,12:SCREEN3:OPEN"GRP:"AS1:DRAW"BM20,60":PRINT#1,"#FIN#";
    :A$=INPUT$(1):BEEP:CLOSE:SCREEN1:GOSUB560:GOTO60
480 '***** PRESENTACION *****
490 COLOR1,10,10:SCREEN1:KEYOFF:WIDTH24:LOCATE0,10:PRINT"AB EL BOSQUE MALDITO
    AB";:LOCATE0,20:PRINT"ALEJANDRO F. BRIZUELA":FORA=0TO1000:NEXT:
    A$="E4RE2REUE2UEU2EU2UE2RER8F2DFD11FD3FDFG3L4HL6GLGL7HUH2LHL2":
    COLOR15,12,2:SCREEN3:LINE(40,60)-(210,160),3,BF
500 DRAW"BM60,120S16C15"+A$:PAINT(125,100),14,15:CIRCLE(130,70),8,1:PAINT(130,
    70),8,1:CIRCLE(155,70),8,1:PAINT(155,70),8,1:CIRCLE(140,120),20,1,0,180*3.14159/
    180
510 PLAY"T68S12M8860 5G4.L16FE-F4.E-FG4.
    FE-L8FGL16FE-DCL11S10M6000E-DCE-DCV15F4.L16E-FG4.
    FE-L8FGL16FE-DE-S13M1000C1","T68V12L2O4GG+AA-V15GG+AA-GG+AA-","
    T68O4V8L11E-DCV10E-DCE-DCV12E-DCE-DCE-DCE-DCE-DCV15E-DCE-DCE-DCE
    -DCE-DCE-DCE-DCE-DCE-DCE-DCE-DCE-DCE-DC4
520 PLAY"T255O7L64S10M6000GE-CO6GE-CO5GE-CO4GE-CO3GE-CO2GE-T32C4",
    "T255L64O6GE-CO5GE-CO4GE-CO3GE-CO2GE-CO1GE-C"
530 IF PLAY(0)=0THENSOUND7,&B00111:SOUND8,16:SOUND9,16:SOUND6,0:SOUND13,
    0:RETURN ELSEVDP(2)=RND(1)*255:GOTO530
540 GOTO540
550 '** TABLA DE PUNTUACION
560 IF SC<SC(6) THEN RETURN
570 COLOR1,10,12:VPOKEBASE(6)+4,&HB2:VPOKEBASE(6)+26,&H4E
580 LOCATE4,10:INPUT"NOMBRE:";N$
590 IF SC>SC(1)THEN FORA=10TO2STEP-1:SC(A)=SC(A-1):N$(A)=N$(A-1):
    NEXT:SC(1)=SC:N$(1)=N$:GOTO650
600 IF SC>SC(2)THEN FORA=10TO3STEP-1:SC(A)=SC(A-1):N$(A)=N$(A-1):
    NEXT:SC(2)=SC:N$(2)=N$:GOTO650
610 IF SC>SC(3)THEN FORA=10TO4STEP-1:SC(A)=SC(A-1):N$(A)=N$(A-1):
    NEXT:SC(3)=SC:N$(3)=N$:GOTO650
620 IF SC>SC(4)THEN FORA=10TO5STEP-1:SC(A)=SC(A-1):N$(A)=N$(A-1):
    NEXT:SC(4)=SC:N$(4)=N$:GOTO650
630 IF SC>SC(5)THEN FORA=10TO6STEP-1:SC(A)=SC(A-1):N$(A)=N$(A-1):
    NEXT:SC(5)=SC:N$(5)=N$:GOTO650
640 IF SC>SC(6)THEN FORA=10TO7STEP-1:SC(A)=SC(A-1):N$(A)=N$(A-1):
    NEXT:SC(6)=SC:N$(6)=N$:GOTO650
650 WIDTH 24:LOCATE0,0:PRINT"$ TABLA DE PUNTUACION $";:LOCATE0,1:PRINTSTRING$(24,
    "R");:FORU=1TO6:LOCATE0,U+1:PRINT"Q ";N$(U);TAB(14);SC(U);TAB(23);"Q";:
    NEXT:PRINTSTRING$(24,"R");:FORU=0TO2000:NEXT:RETURN
```



LOS MEJORES DE INPUT MSX

PUESTO	TÍTULO	PORCENTAJE
1°	SOCCER	17,8 %
2°	KNIGHT LORE	14,5 %
3°	HERO	14,1 %
4°	GREEN BERET	10,4 %
5°	CAMELOT WARRIORS	9,2 %
6°	KNIGHT MARE	8 %
7°	YIE AR KUNG-FU	7,1 %
8°	ALIENS-8	6,5 %
9°	PROFANATION	6,5 %
10°	BATMAN	6 %
		100,0 %

Para la confección de esta relación únicamente se han tenido en cuenta las votaciones enviadas por nuestros lectores de acuerdo con la sección «LOS MEJORES INPUT».

Marzo de 1987



LA AVENTURA AFRICANA

DATOS GENERALES

Título Livingstone supongo

Fabricante Opera Soft

Clase de programa juego

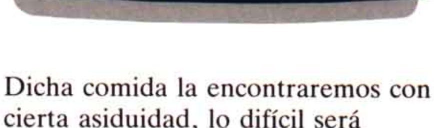
Formato cassette y disquette

CALIFICACION (sobre 10 ptos.)

Originalidad	**8**
Interés	**7**
Gráficos	**6**
Color	**7**
Sonido	**6**
TOTAL	**34**







Estamos en 1871, bajo la identidad de **Henry Morton Stanley**, reportero del «New York Herald», y nuestro objetivo es rescatar y hacer un reportaje a **David Livingstone. Livingstone** es un famoso explorador inglés que en 1866 se fue a África, en su segunda expedición, con el deseo de hallar las fuentes de los ríos Zambeze y Nilo.

En nuestro largo recorrido por tierras africanas deberemos buscar y recoger las cinco piedras sagradas de la tribu de los Ujiji, y llevarlas al templo sagrado de dicha tribu, ya que este poblado es donde suponemos encontrar al **Dr. Livingstone**. Pero por si esto fuera poco, también tenemos que estar atentos a nuestros niveles de hidratación y de nutrición. Para que éstos no bajen más de la cuenta debemos ir comiendo y bebiendo durante nuestra travesía africana. Dicha comida la encontraremos con cierta asiduidad, lo difícil será alcanzarla.

Nosotros conocedores de África, y si no, lo acabaremos siendo, sabemos los peligros que nos esperan: animales salvajes, obstáculos naturales, tribus antropófagas, y para superarlos hemos añadido a nuestro equipaje tres armas y una pértiga. La primera es un bumerán, un recuerdo de uno de nuestros viajes por Australia. Éste es muy eficaz y si sabemos aprovecharnos de su peculiar recorrido nos será muy útil. Nuestra segunda arma es un cuchillo, que lanzamos con gran precisión contra nuestros enemigos. Y la última arma son las granadas, muy potentes y de gran efecto, pero cuidado al lanzarlas, ya que nos pueden alcanzar a nosotros mismos. La pértiga no es un arma pero nos será muy útil, ya que nos permite saltar altiplanos, cascadas, montañas, abismos,...

Quizá algún día tras horas y horas de expedición logremos encontrar al **Dr. Livingstone** y decir la célebre frase: «El **Dr. Livingstone,** supongo».









★★★★★★★★★★★★★★★★★★★★★★★★★

VIVA LAS VEGAS

Cuando **Peter** recibió un telegrama que le notificaba que su tía había muerto y le hacía heredero universal de su fortuna, nunca creyó que la última voluntad de su tía le acarrearía tantos problemas. Y es que para heredar la fortuna de su parienta, tenía que igualar la hazaña que ella había realizado en la década de los años treinta: ganar un millón de dólares en Las Vegas, ¡y en una sola noche!

Lo primero que tiene que conseguir **Peter** es llegar al aeropuerto para coger el avión de las 11:20 hacia Las Vegas (el billete le llegó junto al telegrama con 200 dólares). Él se apresura a hacer la maleta y salir a la calle, pero sus acreedores le persiguen y no le dejarán en paz hasta que no les devuelva todo lo que le dejaron prestado. Al fin una bella señorita le acompañará hasta el aeropuerto, y allí empezará la segunda parte de la aventura. Las prisas por coger el avión y los piratas aéreos, serán las dificultades a superar en esta segunda etapa. Y ahora el **sprint** final, la gran hazaña: lograr un millón de dólares en una sola noche. La ciudad del vicio está a nuestros pies, y los casinos nos ofrecen tres interesantes juegos: el *jackpot, la ruleta americana* o los *dados*; alternando de un juego a otro y con un poco de suerte quizá logremos heredar una fortuna.

DATOS GENERALES

Título La herencia

Fabricante IDEALOGIC

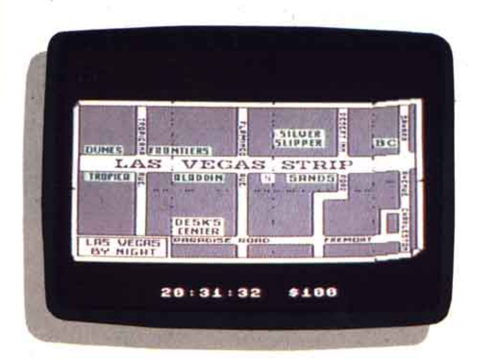
Clase de programa juego

Formato cassette y disquette

CALIFICACION (sobre 10 ptos.)	
Originalidad	9
Interés	7
Gráficos	8
Color	8
Sonido	6
Total	**38**



Ésta es la trama de la original videoaventura de **IDEALOGIC** llamada **La herencia**. Para jugar sólo nos hace falta el *joystick* o las teclas del cursor, esta elección mediante el emplazamiento del cursor permite coger objetos, situarlos en otro lugar, abrir puertas, llamar al ascensor, comprar comida... y esto sin escribir una sola palabra. El programa se encuentra dividido en tres partes: el edificio, el aeropuerto y Las Vegas; al superar cada etapa se nos da una contraseña para acceder directamente a la siguiente. O sea, si queréis descubrir hasta dónde os puede llevar la última voluntad de una peculiar anciana éste es vuestro juego.







★★★★★★★★★★★★★★★★★★★★★★★★★

LA CONQUISTA DEL ESPACIO

Némesis era un planeta tranquilo hasta que sus enemigos decidieron destruirlo por completo. Para salvar a este antiguo pueblo se nos lanza al espacio en el **Warp Ratter**, un prototipo de caza superespacial, al cual podremos ir perfeccionando con más armas de ataque y fuerzas de defensa. Si recogemos las cápsulas de fuerza de nuestros enemigos. Empezamos luchando con un pequeño misil como única arma, gracias a él y a nuestra habilidad logramos destruir a unos cuantos enemigos. Luego recogemos sus



DATOS GENERALES

Título Némesis

Fabricante KONAMI

Clase de programa juego

Formato
cartucho ROM

CALIFICACION (sobre 10 ptos.)

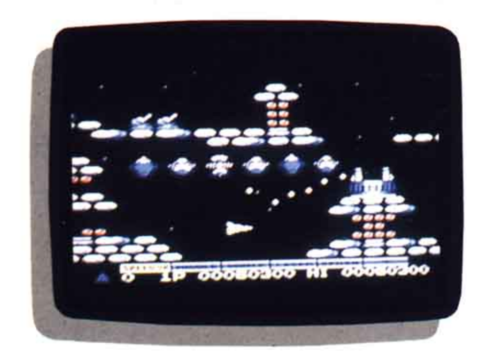
Originalidad	**7**
Interés	**8**
Gráficos	**9**
Color	**8**
Sonido	**8**
Total	**40**



cápsulas de fuerza y reforzamos nuestra nave con un potente láser. El láser es mortífero y rápidamente recogemos otras cápsulas de fuerza que nos permiten sacar a combate el **sidewinder**, una segunda nave que nos sigue como si fuera nuestra sombra. Las siguientes cápsulas las utilizamos para aumentar nuestra velocidad. Una vez somos más veloces que la luz usamos las cápsulas de fuerza para armarnos con misiles aire-tierra. Cuando creíamos ser invencibles e íbamos a crear una barrera de campo, que nos hiciera invulnerables al ataque de los enemigos, nos ha alcanzado un misil y nos ha destruido.

Ya íbamos a poner cinco duros más en la ranura de la máquina cuando nos dimos cuenta de que estaba delante de un MSX con un cartucho de Konami. Y es que la calidad de este programa es casi insuperable. La acción se desarrolla de forma horizontal, y hay un *scroll* de pantalla continua de izquierda a derecha. Nuestra nave se mueve libremente por toda la pantalla y su velocidad es variable, la podemos aumentar hasta ser ocho veces mayor que la inicial, además podemos disparar a discreción sin límite de balas. Al final de cada nivel hay una pantalla de especial dificultad que es necesario pasar para acceder al siguiente nivel. Entre nivel y nivel hay una pantalla de bonus. En un principio contamos con tres naves, pero al final de cada partida, si lo deseamos, podemos continuar desde donde habíamos acabado la anterior, pulsando F5, pero el marcador volverá a cero. Además este juego tiene la posibilidad de que sean dos los jugadores que participen de forma alternativa en las emocionantes y espectaculares peripecias espaciales.

Estoy seguro de que esta novedad de Konami os convertirá en unos intrépidos exploradores del espacio, y las horas de entreno se os pasarán volando.





★★★★★★★★★★★★★★★★★★★★★★★★★★

TELARIUM: LAS AVENTURAS INTERACTIVAS



Philips e Idealogic han lanzado al mercado una serie de **programas interactivos** entre los que destacan títulos como: *Cita con Rama, Dragon World, El Mago de Oz* o *Fahrenheit 451*. Pero ¿qué es un programa



interactivo? Estos programas se distinguen de los juegos normales por el alto porcentaje de participación del usuario. Basándose en una historia real o ficticia, se nos prepara un juego y a partir de ese momento el desenlace final de la historia depende de nosotros, de nuestro poder analítico, del uso que hagamos del entorno y de lo que conozcamos la obra original. Pero estos juegos se basan en best-sellers fáciles de adquirir y en caso de apuro una lectura del libro nos puede ayudar a llegar con éxito al final del programa. Éste se divide en tres partes: la parte visual, el texto explicativo y el sonido. La más importante de ellas es el texto. Saber leer bien lo que nos dicen e interpretarlo correctamente es una de las claves de estos juegos. Después de leer el texto que refleja una acción, un paisaje, una situación o un diálogo entre personajes, nosotros, que representamos el papel de protagonistas en la historia, introducimos una instrucción con la orden precisa que queremos llevar a cabo. Como hemos escogido una de las muchas que se nos presentaban (por ejemplo: podíamos ir al N, S, E o O y hemos elegido ir hacia el E) el desenlace de la historia tomará un rumbo determinado. La imagen es opcional y refleja de forma fidedigna el texto, y a veces sigue a ésta una de las veinte composiciones musicales de cada programa.

Junto con los disquettes en que se encuentra grabado el juego se nos entrega una carpeta ilustrada con un librito de instrucciones y orientación sobre la trama de la historia. En el librito se nos especifica el vocabulario que podemos usar para comunicarnos con el ordenador y también cómo hacerlo. La serie **Telarium** cuenta también con un analizador sintáctico de lenguaje capaz de comprender entre 500 y 700 palabras. De esta manera el usuario puede *hablar* con los distintos personajes. Aparte cada juego tiene un distinto nivel de dificultad que va desde un juego como **El Mago de Oz**, hasta **Perry Mason** basado en la serie escrita por Erle Stanley Gardner. Y es que el poder pedagógico de este tipo de programas es otra de las



características que les diferencia de la gran mayoría de los juegos. El niño que se ponga a jugar con **La isla del tesoro**, basado en la novela de Stevenson, conocerá una obra literaria, aprenderá un vocabulario, razonará ante unas situaciones más o menos ficticias y construirá frases sintácticamente correctas. Y no será un receptor pasivo pues estará actuando como parte activa y dinámica del programa.

Idealogic te acerca la posibilidad de utilizar este tipo de programas, hasta ahora privados del público anglófono. La serie **Telarium** te permite convertirte en el héroe de tu propia aventura; ésta posee multitud de posibles finales, permitiendo que nos encontremos en diferentes situaciones cada vez que juguemos con ella.

Ahora contaremos algunas de las aventuras:

★★★★★★★★★★★★★★★★★★★★★★★★★★★★

FAHRENHEIT 451

Esta aventura interactiva está basada en el libro del mismo nombre de Ray Bradbury, y existe también una versión cinematográfica de esta novela de ciencia ficción, dirigida en 1966 por el cineasta francés François Truffaut y protagonizada por el actor austríaco Oskar Werner y la bella actriz británica Julie Christie.

La historia se desarrolla en Nueva York, pero es una ciudad totalmente diferente de la actual. Un lugar donde los bomberos no acuden para salvar edificios ni vidas, sino para quemar. Y quemar, sobre todo, los libros. **Guy Montag**, el protagonista, es un ex bombero que se ha unido a un movimiento clandestino que tiene como objetivo la conservación de la







literatura universal, y para ello cada miembro del grupo memoriza un libro entero o la obra completa de un autor. Pero como todo movimiento clandestino, éste también es perseguido por la ley y **Montag** deberá ir con cuidado de no ser descubierto por los sabuesos mecánicos o por el cuerpo de bomberos.

Podemos aseguraros con total convencimiento que **Fahrenheit 451** constituye uno de los mejores programas de la serie **Telarium**. Y las ilustraciones presentadas en pantalla son de gran calidad y realismo, desde las fuentes, los hoteles o el hospital. Además es uno de los programas con un nivel de dificultad más elevado y los factores de que depende la vida de **Montag** van desde la memorización de citas literarias, que sirven para comunicarse entre miembros de la resistencia a esta antibibliófila dictadura, hasta llevar una tarjeta de identidad con la información correcta y así poder pasar tranquilamente las inspecciones de los cuerpos de bomberos.

Si aceptas el reto que te propone **Idealogic** te darás cuenta de que nunca hasta ahora una aventura había dependido tanto de ti.

DATOS GENERALES

Título Fahrenheit 451

Fabricante Idealogic-Philips

Clase de programa aventura interactiva

Formato disquette

CALIFICACION (sobre 10 ptos.)	
Originalidad	9
Interés	9
Gráficos	9
Color	8
Sonido	7
Total	**42**

****

DRAGONWORLD





El protagonista es **Amsel,** un científico e investigador y de alguna manera visionario. El **Último Dragón**, un viejo amigo de Amsel, ha sido secuestrado, y le llama a través de la Perla del Dragón para que vaya a rescatarlo. El **Último Dragón** está cautivo en algún lugar del Sur de Simbala. Pero **Amsel** no está solo, cuenta con la ayuda de **Hawkwind,** conocedor de las tierras de Simbala. En un principio estás solo y tras recibir el mensaje de socorro del **Último Dragón** irás a buscar a **Hawkwind**, y juntos emprenderéis un largo viaje en busca de vuestro amigo.

DATOS GENERALES

Título Dragonworld

Fabricante Idealogic-Philips

Clase de programa aventura interactiva

Formato disquette

CALIFICACION (sobre 10 ptos.)	
Originalidad	9
Interés	8
Gráficos	7
Color	7
Sonido	6
Total	**37**

****

EL MAGO DE OZ

Dorothy reposaba tranquilamente en casa de sus tíos, en una granja de Kansas. De repente un resplandor llamó su atención, un ciclón se acercaba a la casa y estaba a punto de absorberla. Tuvo el tiempo justo

para entrar a la casa, coger a **Toto**, su perro, y agarrarse al palo de la cama.

El ciclón se llevó casa, tierra y árboles por los aires. **Dorothy** después de un largo sueño despierta dentro de la casa, todo está patas arriba, **Toto** ladra ante la puerta y se acerca para calmarlo, abre la puerta y se encuentra delante de un paisaje tan bello que el deseo de explorarlo le hace perder cualquier miedo.

¿Te atreves a iniciar un viaje por el **País de Oz** junto a **Dorothy** hasta encontrar el camino de regreso a Kansas?

Al igual que aconteciera con **Fahrenheit 451, El mago de Oz,** antes de pasar a la pantalla de tu microordenador, se constituyó en un sonado éxito cinematográfico en dos ocasiones: en 1939 y 1985.





DATOS GENERALES

Título El Mago de Oz

Fabricante Idealogic-Philips

Clase de programa aventura interactiva

Formato disquette

CALIFICACION (sobre 10 ptos.)

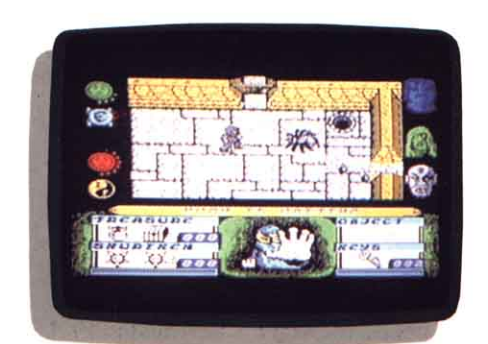
Originalidad	9
Interés	8
Gráficos	7
Color	7
Sonido	6
Total	**37**

★★★★★★★★★★★★★★★★★★★★★★★★★★★★

LAS ARTES MARCIALES

AVENGER es la segunda parte de THE WAY OF THE TIGER. Pero esta vez no se trata de efectuar unos combates cuerpo a cuerpo, con espadas o palos, ahora tenemos una misión más difícil. Hemos de vengar la muerte de nuestro padrastro y recuperar los **papiros de Ketsuin**. Para ello deberemos ir al castillo de Quench Heart y encontrar las llaves que nos abrirán las puertas de las murallas. Una vez dentro lucharemos con los guardianes y si salimos victoriosos quizá logremos recuperar los papiros de las manos del malvado **Yeamon.** Para realizar esta hazaña contamos con la ayuda de **Kwon,** quien responderá a nuestra llamada reponiendo nuestras fuerzas, siempre y cuando no abusemos de su benevolencia. Nuestras armas son los shuricans, la espada mágica, el guante de malla y una palanca. Además de nuestro profundo conocimiento de las artes marciales. A lo largo de nuestro recorrido por el castillo de Quench Heart, encontraremos tesoros, amuletos, hechizos, cuerdas mágicas y los papiros. Cuando estén en nuestra posesión todos los papiros debemos escapar del castillo.

Este programa de **GREMLIN**



DATOS GENERALES

Título Avenger

Fabricante Gremlin Graphics

Clase de programa juego

Formato cassette

CALIFICACION (sobre 10 ptos.)

Originalidad	7
Interés	7
Gráficos	8
Color	8
Sonido	7
Total	**37**



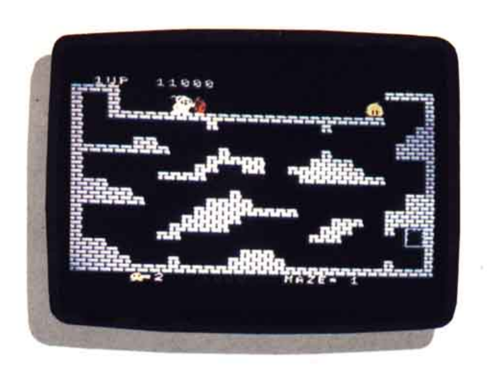


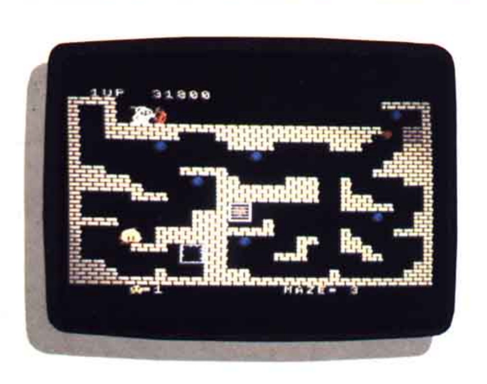
GRAPHICS que tiene como subtítulo THE WAY OF THE TIGER II es una mezcla de los juegos de artes marciales y de los juegos de laberintos. Esta combinación da al programa un dinamismo y un nivel de dificultad que no existían en su predecesor. La pantalla se encuentra dividida en tres partes bien diferenciadas. En la parte inferior se muestran nuestras posesiones, en el lazo izquierdo se nos indica la energía que tenemos tanto nosotros como los guardianes, en el derecho hay un retrato de cada uno de los guardianes y en la parte central se ve a vista de pájaro el lugar del castillo donde nos encontramos. El movimiento del muñeco protagonista es rápido y desenvuelto y los gráficos son de un tamaño menor que en el anterior programa y esto permite que aparezcan más objetos en pantalla. Para completar el programa se le han incorporado dos opciones adicionales como son la de parar el juego o anular la partida.

****

ELIMINA LOS FANTASMAS

Electric Software nos presenta un nuevo programa del estilo de Sweet Acorn. Esta vez el protagonista es **Chack'n** y su misión es rescatar a unos corazones que están cautivos en las profundidades de un viejo castillo. Para lograr este objetivo **Chack'n** deberá vencer a los fantasmas del castillo, para ello cuenta con la ayuda de unas bombas de gran efecto. Pero ¡cuidado! porque tanto pueden matar a los fantasmas como a nosotros mismos. En cada pantalla después de lograr liberar al corazón o corazones que allí se encuentren deberá llegar hasta la salida de esa habitación evitando a los fantasmas y en un corto período de tiempo. Una vez finalizamos con éxito la misión en una pantalla pasamos a otra de mayor nivel, y así sucesivamente. Este sucesivo aumento del nivel crea una cierta adicción hacia el programa y una intriga por ver si seremos capaces de superar el próximo nivel y cuáles serán las siguientes dificultades con las que nos encontraremos. Este factor suple de alguna forma la rutina de la similitud entre las pantallas.





Lo más destacable de este programa es su protagonista, **Chack'n**, un ser alado y patilargo que con su parsimonioso andar llegará a ponernos frenéticos, sobre todo si un fantasma nos está siguiendo.

Una vez logremos alcanzar la salida sanos y salvos y podamos pasar a un nivel más difícil, **Chack'n** lo celebrará saltando de alegría y nosotros, por supuesto, participaremos de su felicidad.

DATOS GENERALES

Título Chack'n pop

Fabricante Electric Software

Clase de programa juego

Formato cassette

CALIFICACION (sobre 10 ptos.)

Originalidad	6
Interés	6
Gráficos	7
Color	7
Sonido	6
Total	32

****

LOS BOINAS VERDES

Este juego de **Konami** completa un producto cinematográfico que tiene como máximo exponente películas como **Rambo,** y se caracteriza por su alto contenido bélico. Pero esta vez el producto nos lo venden los japoneses, aunque los malos siguen siendo los rusos que quedan gráficamente representados mediante unos personajes siniestros y destructores. Éstos han secuestrado a nuestros compañeros, y nosotros bajo el aspecto de un fuerte y omnipotente hombre yanqui vamos a rescatarles, desafiando las infinitas fuerzas del mal. Cada vez que avanzamos hay una masacre y el terreno se llena de cadáveres. Pero la dureza del personaje le hace seguir y conseguir un lanzallamas.



Ahora ya puede enfrentarse a los lanzagranadas y campos de minas del enemigo con mayor garantía de éxito, ya que usando su potente arma, todo lo que se encuentre delante quedará destruido. A lo largo del recorrido el escenario cambiará varias veces, pero la trama siempre será la misma: avanzar y matar. Hasta que al fin, tras matar a media población rusa, liberamos a nuestros compañeros.

Técnicamente el juego es correcto, aunque no es a lo que nos tiene acostumbrados **Konami**. Los gráficos son variados y los escenarios van ligados al tema en que se basa el juego. El movimiento es casi unidimensional, pero también existe la posibilidad de saltar o subir y bajar escaleras, por lo que los esprites que circulan por toda la pantalla toman un movimiento en dos dimensiones, no obstante esta sencillez de movimiento el juego es lento. Por lo que respecta a los gráficos adicionales hay gran variedad de ellos y son de gran originalidad y el sonido se asemeja a una marcha militar.

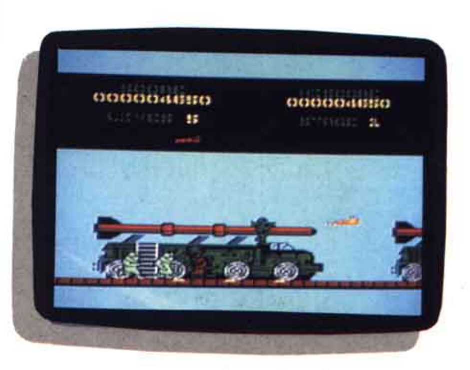
DATOS GENERALES

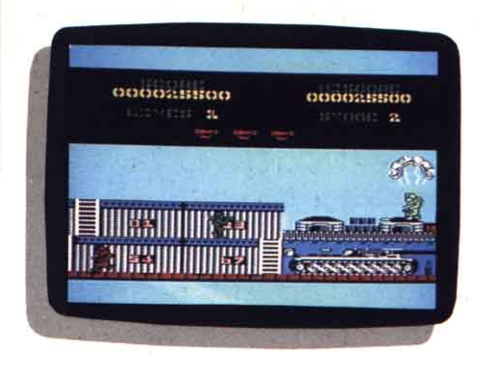
TITULO	Green Beret
FABRICANTE	Konami
CLASE DE PROGRAMA	juego
FORMATO	cartucho ROM

CALIFICACION (sobre 10 ptos.)

ORIGINALIDAD	5
INTERES	6
GRAFICOS	8
COLOR	7
SONIDO	6
TOTAL	32







NO OLVIDES EL TELEFONO... ☎

Cuando, por cualquier motivo, nos escribas, no olvides indicar tu número de teléfono. Así nos será más fácil y rápido ponernos en contacto contigo. Gracias.

EL ZOCO

INTERCAMBIO programas con chicos y chicas de toda España. Poseo entre otros: Profanation, Camelot Warriors, Jack the Nipper, Green Beret, etc. Escribir a: Fco. Javier Manresa. C/ Buenos Aires, n.° 6, 2D. 30011, Murcia.

CAMBIO/VENDO juegos MSX tales como: Samantha Fox, Knightmare, Alien 8, Hero Sky Jaguar, etc. Me gustaría tener: Profanation, Batman, Le Mans, Sorcery, Road Fighter, etc. Angel Matías Rodríguez, c/ Reyes Católicos, 2-17. Telf.: (96) 379 81 19. Xirivella. Valencia.

VENDO Music-Module de Philips para MSX con cinta de efectos especiales, totalmente nuevos y Spectravideo 328 con manuales, grabadores y programas por 40.000 los dos; 22.000 por separado. José Muñoz. Avda. Suecia 4.ª, 30. Valencia - 46010. Teléfono: (96) 369 95 71.

CAMBIO/VENDO programas comerciales MSX en cinta o disco de 3,5. Primeros títulos. También vendo SVI-728 y FLOPPY DISK por un precio irrisorio. Iñaki Fernández Izquierdo, c/ Zamakoa, 7, 5.° dcha. Telf.: (94) 456 33 72. 48960 Galdácano. Vizcaya.

VENDO/CAMBIO cartucho MSX. TRACK & FIELD II de Konami. Luis Sánchez Sabater, c/ Concepción Arenal, n.° 15D, 8.° 2.°. Telf.: (964) 21 92 82. Castellón.

DESEARÍA contactar con usuarios del MSX para cambiar impresiones, etc. y formar un club MSX, preferentemente para Gijón. Angel Muñoz Alcalde, c/ Río Sella, n.° 37, 3.° izq. Cantrueces. Gijón. Asturias. Telf.: 15 41 47.

CAMBIO/VENDO juego Knight Lore. Interesados llamar/escribir a: Domingo Pérez Díaz, c/ Amador Montenegro, n.° 23. Telf.: (986) 41 69 42. Vigo. Pontevedra.

SÁCALE provecho a tu ordenador MSX formando tu propia peña de lotería primitiva con los amiguetes. Llamar a: Miguel Ángel. Telf.: (96) 365 10 52. Valencia. De 6 a 8 tarde.

CAMBIO juegos de Konami (Kung-fu I, Road Fighter, Tenis, etc.) por uno de estos juegos: Taewondo, Hisper Sports III, Kung-Fu II, Green Beret o por un Joystick. David Lorente, c/ Barcelona, n.° 98, 4.° A. Granollers. Barcelona. Telf.: (93) 870 29 13.

VENDO ordenador HIT BIT 75P de 80K, incluyendo cables, 2 manuales, embalaje original, BITCORDER SDC 500 de SONY, 5 libros (lenguaje máquina, programas, etc...), 100 programas (Soccer Konami, Jet Fighter, BC Grog's Revenger, etc...), más de 100 revistas (INPUT MSX, MSX club, etc...) todo por 50.000 ptas. Alberto García Pérez, c/ Bages, n.° 63, 1.° A. Tarrasa. Barcelona. Telf.: (93) 785 51 52.

VENDO/CAMBIO Y COMPRO juegos para MSX. Llamar al 219 67 82. Preguntar por Marc Valero. Barcelona.

VENDO ordenador Philips VG 8000, manuales y cables de conexión (en buen estado) por 22.000 ptas. José Luis Rosa Arroyo, c/ San Luis n.° 37, int. 3. Telf.: (93) 219 56 88. Barcelona.

VENDO ordenador SONY HIT BIT MSX HB-75P 80K, con manual de instrucciones, grabadora y mando de juegos. Todo por: 35.000 ptas. Fco. Muñoz Moreno, c/ Nou de la Rambla, n.° 109 bis. Barcelona - 08004. Telf.: (93) 329 88 89.

INTERCAMBIO programas tanto en cinta como en disco de 3,5". Luis Amado Rego, c/ Puente, n.° 25, 3.° 36500. Lalín. Pontevedra.

CAMBIO/VENDO programas MSX. Poseo todos los de Konami y de Sony; y muchos otros cassettes. Llamar a Alex. Telf.: (93) 332 54 30. Barcelona.

VENDO impresora Philips MSX VW-0020 comprada hace cuatro meses, por cambio de sistema (con manual de referencias y garantía). Precio a convenir. Jorge Muñoz Carmona. Pje. las Celindas, n.° 1, 3.° B. Ecija (Sevilla). C.P.: 41400. Telf.: (954) 83 20 04 de 5 a 8 de la tarde.

BUSCO copión para Philips VG 8020, precio a convenir. Espero ofertas. José Miguel Lambán, c/ Margarita Xirgu, n.° 20, 5.° D. Telf.: 52 02 80. Zaragoza - 50015.

VENDO Camelot Warriors, Martin Basket, Batman, etc... o cambiar por The Way of the Tiger I o II. Llamad de 8 a 9. Telf.: 20 62 39. Juan Carlos González. C/ Fray Luis de León, 15-5.° izq. León - 24005.

CAMBIO Juegos MSX (Ping-Pong, Hyper II, Super Cobra, Boxing, Green Beret... etc.) 80 en total. Juan Manuel García Pedreguera. C/ Juan José Pérez del Molino, 13, 4.° dcha. Telf.: 37 20 19. 39006. Santander. Cantabria.

CAMBIO juegos no originales a convenir. Tennis, Hyper Sports 2 y 3, Track Field 1 y 2, Antartic Adventure. Todos de Konami. Santi y Carlos Arribas. C/ Lasaga Larreta, P-25, 1.° B. Torrelavega. Cantabria.

COMPRO el juego GREEN BERET, preferentemente pasado a cinta. Llamar o escribir a: Enrique García Vega. C/ Lancia, 5-8.° dcha. 24004. León. Telf.: (987) 20 10 81.

CAMBIO últimos programas. Los mejores juegos. Me interesan tipo Arkade o Conversacionales. Thomas C.S. Telf.: (93) 417 69 33. Barna.

VENDO ampliación de 64K Sony HBM-64 por sólo 8000 ptas. Totalmente nuevo. Ignacio Marco Sancho. C/ Cura, 1, 4.° izq. Telf.: 21 64 09. 02001. Albacete.

Fe de erratas:

Por error, en el anterior número de INPUT, algunas de las líneas de programa publicadas figuraban con incorrecciones. Se trata de las siguientes: en el programa *Música y simulación de partituras*, a partir de la línea 22Ø los Ø que figuran entre comillas deben suprimirse por O (de octava).

En el artículo *LOTO MSX*, las líneas 23Ø y 59Ø deben figurar de la siguiente forma:

```
23Ø IF A=M(K,T) THEN GOTO 44Ø
59Ø PRINT MF(L,Ø); TAB(5); MF(L,1); TAB(5); MF(L,2); TAB(5); MF(L,3); TAB(5); MF(L,4); TAB(5), MF(L,5).
```